F I D

Fuzzy Application Software

Users Manual
Demo Version 1.6

Windows based Fuzzy Development Tools
for ADSP 21xx

(c) 1993 - 97 FDG-Systems, Dipl.-Ing. Norbert Exler, Vienna Austria
All rights reserved

PROGRAM LICENSE AGREEMENT

made between

hereafter called the customer

and

M. Bammer and N. Exler, trading as FDG-Systems hereafter called the owner
Leystrasse 48/25
1200 Vienna
Austria - Europe
100043.151@CompuServe.com

BY OPENING THE SEALED DISK PACKAGE, YOU INDICATE YOUR ACCEPTANCE OF THE FOLLOWING FID FUZZY APPLICATION SOFTWARE LICENCE AGREEMENT. IF YOU DO NOT ACCEPT OR AGREE TO THESE TERMS YOU MAY, WITHIN FIFTEEN DAYS, AT YOUR EXPENSE RETURN THE UNOPENED DISK PACKAGE AND ALL ACCOMPANYING ITEMS TO THE OWNER FOR A FULL REFUND.

1. The owner hereby grants the customer a non-exclusive licence to use the FID Fuzzy Application Software on a **single** computer workstation. The FID Fuzzy Application Software shall not be shared between multiple computer workstations. Each computer workstation must have its own separately-licenced software. Ownership of the FID Fuzzy Application Software Package is **not** transferred to the customer.

2. The customer may copy the FID Fuzzy Application Software, in whole or in part, only for back-up and archival purposes. **No more than one copy** may be in existence at any time. That copy must include any and all confidential, proprietary, and copyright notices or markings contained on the original FID Fuzzy Application Software in readable format. That copy becomes automatically the property of the owner and is subject to this agreement. You may physically transfer the program from one computer to another over a network but the customer may not install the FID Fuzzy Application Software on a network to be used by more than one workstation at any time.

3. The customer may transfer the FID Fuzzy Application Software provided that
   (i) this software licence agreement is transferred with the FID Fuzzy Application Software,
   (ii) the transferee fully accepts the terms and conditions of this agreement, and
   (iii) all complete or partial copies of the FID Fuzzy Application Software, including copies on data storage devices are also transferred (or destroyed).

   In case of a licence transfer, the original disks shall be sent back to the owner with the full name and address of the new customer and all backup disks have to be destroyed. The so named new customer will receive a new licenced version of the FID Fuzzy Application Software. Until the delivery of the new version the customer is allowed to use the old licence. After receiving the new licence the customer shall recompile all written applications. The customer of the software may not sub-licence, sell, lend, rent, or lease any portion of the FID Fuzzy Application Software.

4. The customer may not alter the digital configuration of the FID Fuzzy Application Software, or solicit others to cause the alteration of the digital configuration of the FID Fuzzy Application Software, so as to change the FID Fuzzy Application Software

5. The customer agrees to use all reasonable efforts to ensure that persons employed by the customer or under the customer´ s direction and contro l who come into contact with the FID Fuzzy Application Software abide by the terms and conditions of this agreement (including, without limitation, not knowingl y permitting anyone to use any portion of the FID Fuzz y Applic ation Software for the purpose of deriving its sourc e code). If the customer becomes aware that the FID Fuzz y Application Softwa re is being used by such persons in a manner not authorised by this Agreement, the customer shal l immediately use all reasona ble efforts to have such unauthorised use immediately cease. The customer shall notify the owner in writing of the unauthorised use, if it continues for a n unreasonable period after the customer becomes aware of it.

6. The FID Fuzzy Application Software is protected by copyright and other proprietary rights of the owner. The customer may be held directly responsible by the owner, for acts done by him or on his behalf relating to the FID Fuzzy Application Softwar e which are not authorised by this agreement.

LIMITED WARRANTY

The owner hereby warrants that, for a period of 60 (sixty) days from the date of delivery to the customer, the FID Fuzzy Application Software shall materially conform to the performance defined in the documentation of the FID Fuzzy Application Software relating thereto (manuals, guides, and computer-aided instructions). The owner does **not** warrant that the operation of the FID Fuzzy Application Software will be uninterrupted or error-free or that the FID Fuzzy Application Software will meet the customer's requirements or will operate in the combinations chosen by the customer. The customer's sole and exclusive remedy for failure of the FID Fuzzy Application Software to substantially conform to the performance defined in the applicable documentation is for the customer to return the FID Fuzzy Application Software to the owner whose sole obligation shall be, at its option, to either provide the customer with FID Fuzzy Application Software conforming to the express warranty above or to refund to the customer the licence fee paid therefore. In the event of a refund the customer shall send the original disks and documentation back to the owner, free of any charges for the owner and promptly destroy all copies of the FID Fuzzy Application Software in its possession, if any, and confirm in writing to the owner that all copies were destroyed. In order to claim the guarantee services, the customer must provide an exact description of the computer system used, as well as the conditions under which the faults appeared. After receiving the disks and the confirmation, the refund will be given.

The owner grants no other warranties on the FID Fuzzy Application Software and any liability of the owner for breach of any other warranty or condition, express or implied, whether by statute or otherwise, including any liability for indirect or consequential loss or damage with the use of the FID Fuzzy Application Software is hereby expressly excluded. The owner further disclaims any warranties or representations made by persons other than the owner (Including but not limited to the owner Software distributers and dealers).

The owner may at its sole discretion offer the customer up-dates of the FID Fuzzy Application Software. The owner retains the right to require payment of an additional fee for such up-dates. The customer may, of course, decline such up-dates.

The customer is solely responsible for the selection of the FID Fuzzy Application Software to achieve the customer intended results.

This Licence is effective until terminated. The customer may terminate it at any time by sending back the original disks and documentation at the customers expense and destroying the copies, modifications and merged portions in any forms. It will also terminate if the customer fails to comply with any term or condition of this Licence Agreement. The customer agrees upon such termination to send back at the customers expense to the owner the original disk and documentation and destroy any copies, modifications, and merged portions in any form.

This software licence agreement comes under Austrian jurisdiction and Austrian law is applicable.

Trademarks

CompuServe is a registered trademark of CompuServe, Inc.
EZ-LAB is a registered trademark of Analog Devices, Inc.
Hewlett-Packard, Laserjet, HP, and PCL are registered trademarks of Hewlett-Packard Company.
IBM, PC-DOS and OS/2 are registered trademarks of International Business Machines Corporation.
MatLab is a registered trademark of The MathWorks, Inc.
Novell and Netware are registered trademarks of Novell, Inc.
NT is a registered trademark of Norhtern Telecom Limited in the USA and other countries.
Windows and MS-DOS are trademarks of Microsoft Corporation.

Other trademarks are the property of their respective holders.

Printing History:

Preliminary release	6/94
Reprint Preliminary release	8/94
Final release 1.4	12/94
Electronic release 1.5	2/97

Thanks to
Paula Goodwin
and
Assumpta Petit
for their
patience and help.

1. Contents

PROGRAM LICENSE AGREEMENT 1

LIMITED WARRANTY 4
Trademarks 6

1. Contents 8

Preface 11

2. Overview 13
2.1 Just one other important thing 13
2.2 About the user manual 14
2.3 Manual System Conventions 15
2.4 The FID Software Package 16
2.5 Program extensions used by this program 17
2.6 System requirements 18

3. How to install the FID Program 19

4. Description of the dialogs - Menus: 22
4.1 The Toolbar 22
4.2 Fuzzy Resource Status View 24
4.3 Fuzzy Resource Editor 25
4.4 LingVar Editor 27
4.5 Fuzzy Set Name Editor 28
4.6 Fuzzy Set View 29
4.7 Rule Resource Editor 30
4.8 Rule Definition - Karnaugh Veitch 31
4.9 Rule Definition - Expert type 33
4.10 Defuzzification 34
4.11 Fuzzy Map Dialog 35
4.12 Define Inputs NOT used for map 36
4.13 Fuzzy Map View 37
4.14 PID input dialog 39
4.15 Linear Plant Definition - Zero-Pole representation . . . 40

(c) 1993-97 FDG-Systems N. Exler

4.16 Zero-Pole-Gain & Deadtime input 41
4.17 Linear Plant Definition- Numerator-Denominator . . . 42
4.18 Numerator-Denominator & Deadtime input 43
4.19 Linear Plant Definition - Multiple steps 44
4.20 Multi-Step definition 44
4.21 Non-linear Plant Definition 45
4.22 Simulation Definition 46
4.23 Connect Fuzzy Inputs 47
4.24 Connect Disturbance 50
4.25 Simulation View 51
4.26 Save simulation data 52
4.27 SDF Definition 53
4.28 Convert Fuzzy to ADSP-Code 54
4.29 About FID 55

5. Fuzzy-Logic on an ADSP-21xx 56
5.1 Features 56
5.2 Function List 57
5.3 Specifications 58
5.3.1 Fuzzification 58
Memory Model for the fuzzification interface: 61
5.3.2 Inference mechanism 62
Memory model for the inference mechanism in loop-mode 66
Memory model for the inference mechanism in sequential-mode 67
5.3.3 Defuzzification 68
Memory model for the defuzzification 71
5.3.4 Tuning 74
Input variable tuning 74
Output variable tuning 75
5.3.5 Resolutions and Formats 75
Input values and scaling factors 75
Membership functions for the input values 76
Membership values (grades) 76
Membership functions (singletons) for the fuzzy output 76

(c) 1993-97 FDG-Systems N. Exler

Crisp output values 76
5.3.6 Implementation of the generated ADSP-21xx source code in your specific application 77

6. Examples running on EZ-LAB board 85
6.1 Design of linear and non-linear static fuzzy-elements . 85
6.2 Design of a two-level fuzzy controller 94
6.3 Design of a simple fuzzy PI-controller compared to a classical PI-controller 100
6.4 Design of a simple fuzzy PID-controller 108

7. Introduction to Fuzzy Logic 113
7.1 Overview 113
7.2 Fundamentals of Fuzzy Logic 113
7.2.1 Fuzzy sets and membership functions 114
7.2.2 Shape of membership functions 119
7.2.3 Set operators 121
7.3 Structure of a fuzzy system 124
7.3.1 Fuzzification 125
7.3.2 Inference and Composition 127
7.3.3 Defuzzification 130
7.4 Fuzzy Control 132
7.4.1 Control structure 132
7.4.2 Design steps of a fuzzy controller 132

8. Bibliography 134

Index 136

(c) 1993-97 FDG-Systems
N. Exler

Preface

Fuzzy logic was first introduced in 1965 by Lofti A. Zadeh but it was ridiculed and not taken seriously by many of Zadeh's contemporaries. The following years, therefore, saw limited work on fuzzy logic and only a small number of people concentrated on developing the first applications of it. Inspite of this though, the first fuzzy controller was realized by Mamdani and his colleagues in the 1970's and was successfully used for the controller of a rotary cement kiln in Denmark. At the end of the 1980's and beginning of the 90's Japanese engineers showed the world that they had taken fuzzy logic very seriously indeed and were able to present numerous applications of fuzzy logic, many of which were to be found in household appliances and consumer articles. The world woke up and suddenly there was a worldwide fuzzy boom !

(c) 1993-97 FDG-Systems N. Exler

Many human decision processes can be reproduced relatively accurately with fuzzy logic. This means it is possible to reproduce those processes carried out by humans for which there is no control model. This reproduction is generally not very difficult. More problematic, however, is the implementation of fuzzy logic onto hardware, especially for cheap and fast single processor solutions.
There are some special fuzzy processors but these are mostly employed as just coprocessors and are therefore not likely to become standard. Regular microprocessors mostly lack the necessary speed for fuzzy logic

applications. As fuzzy algorithms are principly complex and systematic signal processing problems, the signa l processor is the best suited hardware for dealing wit h them. The Analog Devices ADSP 21xx series has lon g since not just processed signals but also has a timer , serial ports, host interface ports, ADC/DAC along wit h much more on chip. ADSP's, therefore cover both fuzzy logic as well as regular signal processing tasks such a s FFT, digitial filters, PID-controllers and so on. Thi s means the combination of the processor family ADSP - 21xx, with one development tool and varied softwar e packages along with their libaries can be used in a ver y wide range of development areas.

The FID Application Software is one of the 3rd part y software packages for the ADSP-21xx family tha t supports Fuzzy development on DSP's. The FI D Application Software is an easy to use Microsof t Windows based application software for developing , writing and testing fuzzy programs based on the ADSP- 21xx system. This makes you independent of specia l fuzzy controllers.

Vienna, August 1994

(c) 1993-97 FDG-Systems N. Exler

2. Overview

Welcome to the world of fuzzy logic!
Thank you for choosing the FID fuzzy applicatio n software. We are sure, that you will enjoy working with this program.

A solid, sound working useable software is a goo d software. Its documentation should be comprehensiv e and clear and we have endeavored to make ours so. But everyone makes mistakes! Considerable time and effort have been spent in the development of this product. I f you are happy with the software, we would like to hea r from you. If you find any mistakes in our software o r think a part of the documentation could be more clearly written, we would *really* like to hear from you. For your convenience, a reply form is provided at the back of this manual or just send us an eMail!

(c) 1993-97 FDG-Systems N. Exler

2.1 Just one other important thing

We think this software is not expensive as it is at a price that everyone can afford. In fact, we think the price o f the software is cheap compared to other products. Thi s competitive price means people can buy it and use i t legally. Although we do not think that you will give the software to someone else, we know that it happen s sometimes. Please read the license agreement carefull y and fulfil your side of the bargain and remember ever y

software you install should be licensed! This is the only way we can go on developing cheap software for you!

2.2 About the user manual

The FID documentation consists of three parts:

- the user manual (which you are now reading),
- the on-line help via F1 when you start the program , and
- the README.TXT file.

(c) 1993-97 FDG-Systems N. Exler

The user manual is a comprehensive description of what the FID software package is and what the included additional tools and programs are. Its chapters are briefly described as followed:

Chapter 1: Table of Contents

Chapter 2: Contains some general useful hints about the program and the manual.

Chapter 3: Describes how to install the FID application software and how to start the first example

Chapter 4: Tells you more about FID. How you can use it and what the different menus do. It starts with a small, simple example which you will find again in several places in this documentation.

Chapter 5: Explains how you can link the generated source code to a program.

Chapter 6: Documentation of the sample programs for the EZ-LAB. Here it will be explained what

the sample programs for the EZ-LAB are , what they do and how you can downloa d and start them on the EZ-LAB

Chapter 7: For all who do not know a lot about fuzz y logic. It is a short introduction to fuzz y logic, but it should not be understood as a n exhaustive tutorial. We have tried to writ e clearly and make it easy to read, and w e have also included s ome mathematics where we think it might be useful. For mor e information about fuzzy logic you shoul d read one of the references included at th e end of this chapter.

Chapter 8: Bibliography

If you are a beginner in Fuzzy Logic you should rea d chapter 7 first. Then continue with chapter 2.

(c) 1993-97 FDG-Systems N. Exler

2.3 Manual System Conventions

To ease the reading of the manual all parts of the manual share several common same name and displa y conventions, and defined terms:

- ☞ The "☞" subheader (like this one) marks highlights, features or technical points.
- ☞ The "☺" subheader gives hints for handling th e program
- ☞ Every windows button or special keyboard ke y which you should press will be written in brackets .

For example **<*OK*>** means that you should press the OK button of the actual dialog box, or **<*enter*>** means that you should press the enter or return key of your keyboard.

☞ Whenever you should type something to perform a function or enter data information it will be written in "typewriter-style". For example:

```
L
```

indicates that you should type the "L" key, while

```
first.flc <OK>
```

instructs you to type "first.flc" and then press the OK button.

(c) 1993-97 FDG-Systems N. Exler

2.4 The FID Software Package

Please be sure the following items are included:

- One 3½" diskette containing the FID program and additional tools.
- One FID License Agreement, found inside the front cover of this manual. It is important to read and understand the Agreement. Check also if the license is correctly written.
- One License Registration Card (at the end of this manual). The serial number of your FID program is printed on the license agreement. When placing a call to FID technical support, please have this number available and include it in all mail you send. If you wish to receive further information about other programs and newer releases, fill it out and send it back to FDG-Systems.

- One FID documentation

If this package does not contain all these items, pleas e eMail to FDG-Systems.

2.5 Program extensions used by this program

A short description of commonly used file extensions:

A	Subroutine libraries
DEF	Files which contain internal definitions
DLL	These files are Dynamic Linked Libraries, called by programs running under Windows and contain executable code.
DSP	Files with ADSP source code text file, generated by the program.
EXE	Files are executable and can be started either under DOS or Windows
FLC	Files contain the Fuzzy resources such as linguistic variables, rulebase, defuzzification, ..
HLP	Help files for the FID application software
INI	Initialization file of the program
LIB	Library file
OBJ	Object files generated by the ADSP compiler
PID	Definition of conventional PID controller
PLH	Definition of steps used for simulation
PLN	Plants defined by means of Numerator/Denominator representation
PLT	Plants defined either by Numerator /Denominator or Zero/Pole means.
PLZ	Plants defined by means of Zero/Pole

(c) 1993-97 FDG-Systems N. Exler

representation
SYM Symbol files used for the ADSP simulator
TXT Files which contain readable text.

2.6 System requirements

To run FID Fuzzy Application software you must fulfi l the following needs:

IBM AT, PS/2 and all 100% compatible machines,
4MB RAM,
Harddisk with 10MB free memory, and
Windows 3.1 or a compatible program

We suggest in order to get an acceptable performance at least 486-33MHz.

☞ If you want to use the program in a network, ask for an FID network license.

(c) 1993-97 FDG-Systems N. Exler

3. How to install the FID Program

Before you continue make a copy of the disk by means of DOS´ diskcopy; Write the name of the software, the license number and the copyright on the label, and keep the original diskette in a safe place. Continue the installation with the copy you have made. The diskette is not copy-protected, but every diskette has an unique serial number.

Start Windows, insert the copied diskette into your disk drive A: and start the installation program through the Program Manager File/Run `a:setup` <***enter***>.

The installation program will ask you where to install the FID Fuzzy Application Software. That´s all you have to do. After some minutes the installation is finished and the program ready to use.

☞ Although we are sure that we deliver this software package free of viruses, we suggest that you check the FID Software with a virus scanner if one is available **before** you start it the first time.

During the installation the following directories will be created:

[path]\FID contains the program and all additional tools needed for running the FID software

[path]\FID\SAMPLES contains all sample programs

(c) 1993-97 FDG-Systems N. Exler

[path]\FID\EZ-LAB	contains all files needed for running the samples on the EZ-LAB
[path]\FID\FUZLIB	contains the fuzzy library files

The following files should be located in the directories:

in [path]\FID:

FID.EXE	the main program
FID.HLP	the help file
README.BAT	additional information to the program
README.LAB	additonal information for the EZ-LAB samples
README.WIN	additional information for the FID Application Software

(c) 1993-97 FDG-Systems N. Exler

in [path]\FID\SAMPLES:

```
FIRST.FLC, FPID.FLC, IMAGCODE.FLC, KV1I1O.FLC
KV2I1O.FLC, SIMTEST.FLC, FPI_25.FLC, FPI_9.FLC
STAT_FS.FLC, STAT_TNC.FLC, TRUCK.FLC
TWOP_H.FLC, TWOP_WH.FLC
PT2.PLN, PT1.PLN, PT2S.PLN, PT2_EX3.PLN    ,
PDT2_T.PLN
PT2.PLZ
STOER.PLH, STOERM.PLH, TWO.PLH
PI_EX3.PID,  PI.PID,  PID.PID
```

in [path]\FID\EX1:

```
STAT_FS.DSP, STAT_FS.EXE, STAT_FS.SYS
STAT_INF.DSP, STAT_INF.EXE, STAT_INF.SYM
STAT_KI.DSP, STAT_KI.EXE, STAT_KI,SYM
STAT_TNC.DSP, STAT_TNC.EXE, STAT_TNC.SYM
```

in [path]\FID\EX2:

```
TWOP_H.DSP, TWOP_H.EXE, TWO_H.SYM
TWOP_PT1.DSP, TWOP_PT1.EXE, TWO_PT1.SYM
TWOP_WH.DSP, TWOP_WH.EXE, TWO_WH.SYM
```

in [path]\FID\EX3:

```
FIRST_H.DSP, FIRST_H.EXE, FIRST_H.SYM
FIRST_S.DSP, FIRST_S.EXE, FIRST_S.SYM
FIRST_R.DSP, FIRST_R.EXE, FIRST.SYM
```

in [path]\FID\EX4:

```
FPID_H.DSP, FPID_H.EXE, FPID_H.SYM
```

in [path]\FID\EZ-LAB:

```
EZ_LAB.ACH, EZ_LAB.SYS
```

in [path]\FID\FUZLIB:

```
FUZLIB.A, FUZTOOLS.LIB, FUZLIB.TXT
```

(c) 1993-97 FDG-Systems N. Exler

4. Description of the dialogs - Menus:

4.1 The Toolbar

Under the main menu is a tool bar with several buttons. The different buttons mak e the access of commonly used funtions and menus very fast. Here is a short description:

(c) 1993-97 FDG-Systems N. Exler

Save fuzzy resource to disk (File menu "Save")

Open an existing fuzzy resource file (File menu "Open")

Open a new fuzzy resource (File menu "New")

Show fuzzy control map (Fuzzy menu "Fuzzy control map

Defuzzification method (Fuzzy menu "Defuz. Definition")

Rule definition (Fuzzy menu "Rule Definition")

Linguistic variable definition (Fuzzy menu "Ling. Var. Definition")

Start simulation (Simulation menu "Start")

Define simulation (Simulation menu "Definitions")

Linear Plant definition (Plant menu "Linear")

PID controller definition (PID menu "PID Definition")

Save simulation data to disk (Simulation menu "Save Data'

Display actuaction (Simulation menu "Show Actuaction")

Show error in simulation view(Simulation menu "Show Err

Show step in simulation view (Simulation menu "Show Ste

(c) 1993-97 FDG-Systems N. Exler

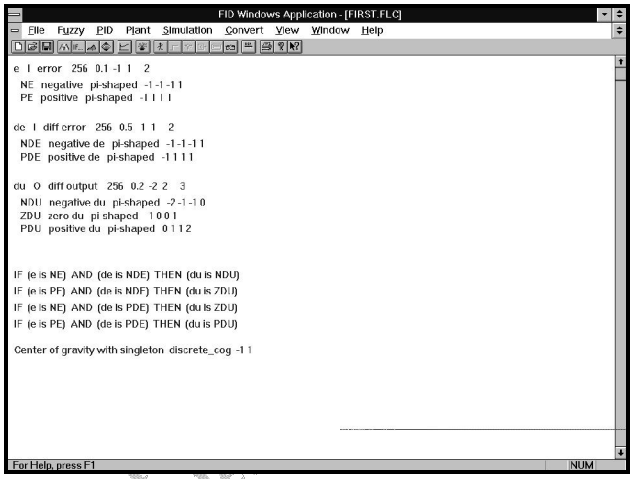
Give Help to a specified object

Some information about FID (Help menu "About FID")

Print the actual selected View/Resource (File menu "Print"

Convert fuzzy resource to ADSP source-code (Convert me

4.2 Fuzzy Resource Status View


FID Windows Application - [FIRST.FLC]
File Fuzzy PID Plant Simulation Convert View Window Help
e I error 256 0.1 -1 1 2
NE negative pi-shaped -1 -1 -1 1
PE positive pi-shaped -1 1 1 1
de I diff error 256 0.5 1 1 2
NDE negative de pi-shaped -1 -1 -1 1
PDE positive de pi-shaped -1 1 1 1
du O diff output 256 0.2 -2 2 3
NDU negative du pi-shaped -2 -1 -1 0
ZDU zero du pi shaped 1 0 0 1
PDU positive du pi-shaped 0 1 1 2
IF (e is NE) AND (de is NDE) THEN (du is NDU)
IF (e is PE) AND (de is NDE) THEN (du is ZDU)
IF (e is NE) AND (de is PDE) THEN (du is ZDU)
IF (e is PE) AND (de is PDE) THEN (du is PDU)
Center of gravity with singleton discrete_cog -1 1
For Help, press F1
NUM

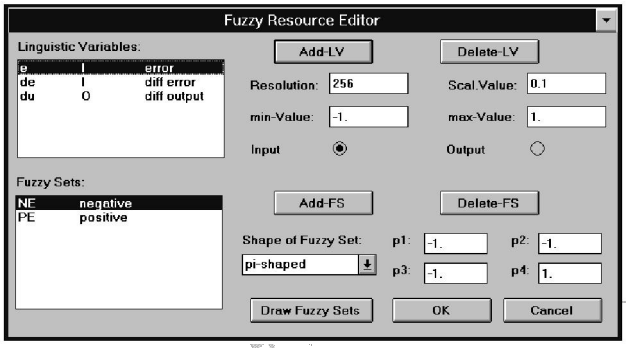
Shows the state of the fuzzy resource file. You can se e the linguistic variables, fuzzy sets, rules, and th e defuzzification method in a short form.

An empty fuzzy resource file shows the following three lines:

No linguistic variables defined!
No rules defined!
No defuzzification defined!

If one of the three parts above is missing the appropriate indication will be shown in the Status View.

4.3 Fuzzy Resource Editor



This dialog window consists of two halves:

- the upper half shows you the **Linguistic Variables**, the lower half the **Fuzzy Sets**.

On the top left of the window a list box shows all linguistic variables already defined. Each line has three entries: the abbreviation , the state (if it is an input or an output), and the name of the linguistic variable.

If a linguistic variable in the list box is selected, the resolution, the scale value and the minimum and maximum of the selected linguistic variable is shown to the right of the list box. Whenever a linguistic variable is selected you can change their properties.

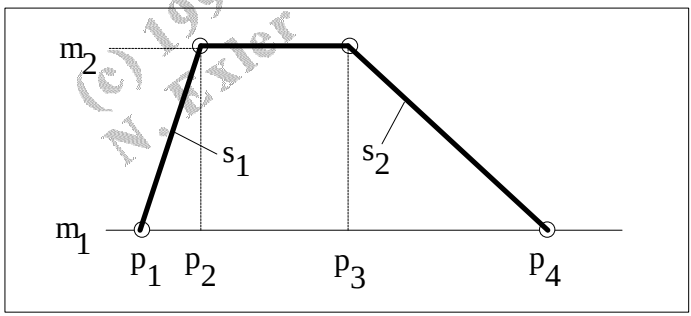
To **add** a new linguistic variable, just press the ***<Add-LV>*** button and the **LingVar Editor** will appear.

To **delete**, select the linguistic variable and press the ***<Delete-LV>*** button.

The lower half always shows the membership functions

(if there are any) of the current selected linguisti c variable. Right of the list b ox you can see and change the properties of the actual highlighted membershi p function. There are three different shapes available. **Pi-shape**, which will always be used when you generate an **ADSP source-code**. **Gauss-shaped** and **cos-shape** membership functions are also available, but only fo r simulation purposes. It is not possible to use gauss - shaped and cos-shaped membership functions in th e ADSP source code, because there is no significan t difference in the behaviour of the fuzzy elements. Th e four points **p1**, **p2**, **p3**, **p4** define the curve completely as follows:

(c) 1993-97 FDG-Systems N. Exler

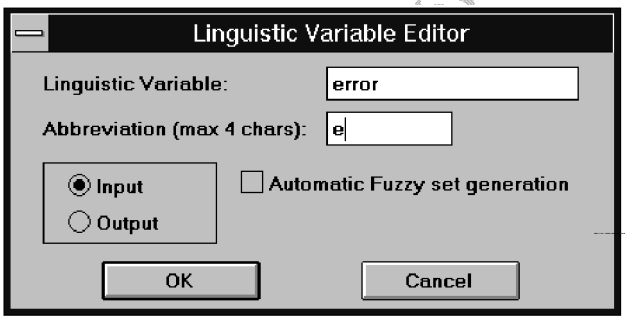


init order: p1 , m1 , p2 , s1 , p3 , m2 , p4 , s2

To **add a new fuzzy set**, just press the ***<Add-FS>*** button and the **Fuzzy Set Name Editor** will appear.
To **delete**, select the Fuzzy Set and press the ***<Delete-FS>*** button.

Whenever you press the ***<Draw Fuzzy Sets>*** button or change the shape of a membership function all membership functions of the current linguistic variable will be shown in a **Fuzzy Set View**.

4.4 LingVar Editor

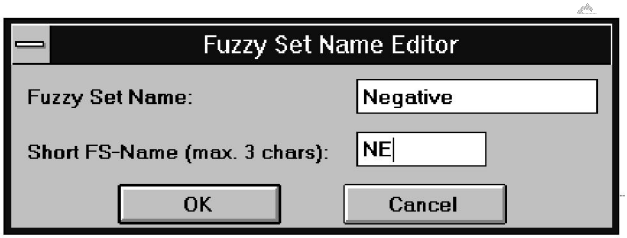


Appears when you press the ***<Add-LV>*** button of the **Fuzzy Resource Editor**. You can enter the name, the abbreviation and the state (input or output) of the new linguistic variable. Ensure that you enter the right name and abbreviation, as you are not able to change it later! If you want to change the abbreviation and/or the full name of the linguistic variable, you must delete the linguistic variable and define it again. The other properties can be entered and changed later inside the **Fuzzy Resource Editor**.

To ease the design of your fuzzy sets it is possible to check the Automatic Fuzzy set generation box. If the checkbox is marked 5 fuzzy sets are automatically

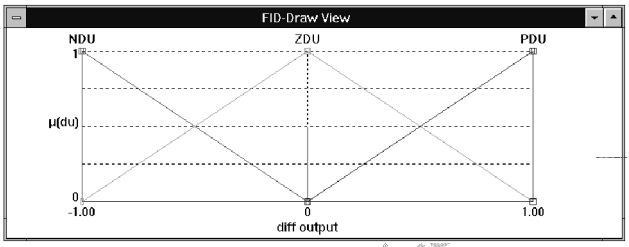
generated when you press the OK button.

4.5 Fuzzy Set Name Editor



The Fuzzy Set Name Editor appears whenever you press the ***<Add-FS>*** button of the **Fuzzy Resource Editor**. You can enter the fuzzy set name and the abbreviation of the new fuzzy set.
Ensure that you enter the right name and abbreviation, as you are not able to change it later! If you want to change the abbreviation and/or the full name of a fuzzy set, you must delete the fuzzy set and define it again. The other properties can be entered and changed later inside the **Fuzzy Resource Editor**.

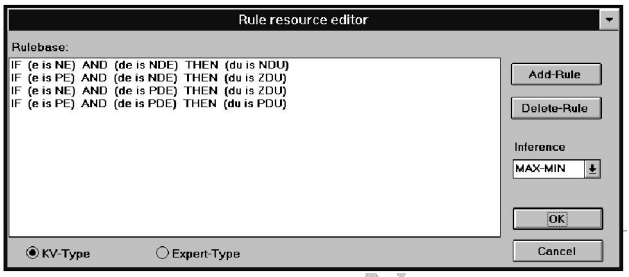
4.6 Fuzzy Set View



The Fuzzy Set View, which shows the currently defined fuzzy sets, appears either when you press the ***<Draw Fuzzy Set>*** button in the Fuzzy Resource Editor or whenever you change the shape of a fuzzy set.
When this view is active you can change the fuzzy sets (the four points p1, p2, p3, p4) simply by locating the cursor inside the box representing one of these points, pressing the left button and moving the point horizontally to a place you want.
If there is more than one point at the same location and you want to select the second one, press the SHIFT button and hold it when you press the left mouse button.
Three different sizes for the fuzzy sets are available and can be changed in the Fuzzy Set View. By double clicking the right mouse button you can enlarge/reduce the fuzzy sets.

(c) 1993-97 FDG-Systems N. Exler

4.7 Rule Resource Editor


Rule resource editor
Rulebase:
IF (e is NE) AND (de is NDE) THEN (du is NDU)
IF (e is PE) AND (de is NDE) THEN (du is ZDU)
IF (e is NE) AND (de is PDE) THEN (du is ZDU)
IF (e is PE) AND (de is PDE) THEN (du is PDU)
Add-Rule
Delete-Rule
Inference
MAX-MIN
OK
Cancel
KV-Type
Expert-Type

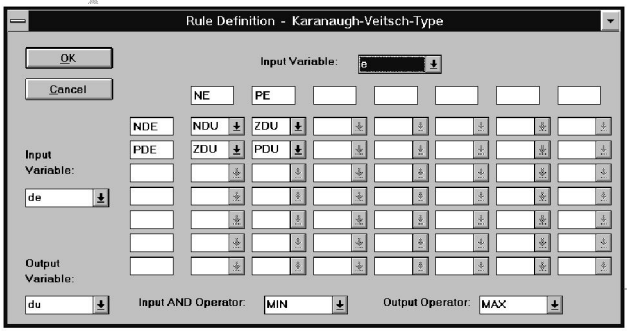
This dialog window shows you all defined rules for th e actual selected fuzzy resource file.
Every Rule starts with an "IF" statement, and alway s consists of at least one condition (antecedent) and on e conclusion (consequent) which are written in brackets. If there is more than one condition in a rule thes e conditions are linked together with an operator between them. The operator between the conditions is defined in the Rule Definition Dialog.
The conclusion of the rule sta rts with a "THEN" and may also consists of more than one part.
The **inference operator** can be selected by the comb o box in the Rule Resource Editor and defines the method of inference. This method of inference links all output s which are fired by several rules, to the same fuzzy set at the output together with this operator(e.g. MAX-MI N inference, the first operator links the same output fuzz y sets together).

To **create a new rule** you have to make sure no rule is selected, select the rule type by checking one of the radio buttons under the list box and press the ***<Add Rule>*** button. The according Rule Definition dialog will appear and you can enter the new rule.

If you want to **change a rule**, select that rule and press the ***<Edit Rule>*** button. According to the rule type the appropriate Rule Definition dialog will appear.
To **change the rule type** of a rule, select the rule and check the radio button you want.
To **change the inference type** select the one you want in the combo box.

(c) 1993-97 FDG-Systems Exler

4.8 Rule Definition - Karnaugh Veitch



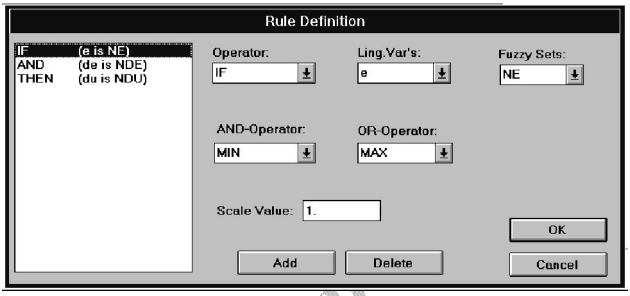
You can define the rules by means of a schedule. Read it as follows:
The linguistic variable at the top is the first input variable and will be the first part of the first condition. Under the

linguistic variable the first row shows all defined fuzz y sets of that linguistic variable which will become th e second part of the first condition.
On the left side you find the same system for the second input which will become the second condition.
Each combo box has all fuzzy sets of the selected output variable and a "**no rule**". Now just select the conclusions of each rule.
Select the operator which should be taken for the AN D operation of the two conditions, and the operator of th e output variable. These Operators will be taken for every rule.
If you have selected a rule before pressing the ***<Edit Rule>*** button, the rules already defined will be shown in the related combo boxes.

You can only create rules consisting of one or two inputs and one conclusion.

(c) 1993-97 FDG-Systems N. Exler

4.9 Rule Definition - Expert type



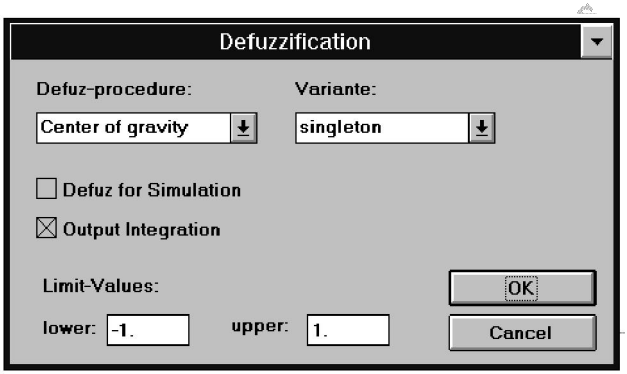
(c) 1993-97 FDG N. Exler

With this type of definition you can freely design the rule. **But be aware of what you do, as the system just follows the rules you define**. No checks of plausibility will be made, and so some rules can cause nonsensical results.

The system of entering the rules is as follows:
Each Line in the list box shows one condition or conclusion of the rule.
Every rule must start with an IF part (the first line) and must have at least one conclusion.
If you want to **enter a new rule part**, select the Operator, the LingVar, the Fuzzy Set and press the ***<Add>*** button. The new part will be added, either before the current selected part or at the end of the rule part list.
To **delete a rule part** select it and press the ***<Delete>*** button.
In the combo boxes for the type of operators, you can select the appropriate operators for the AND and OR - operator for the actual rule. The scale value for each rule

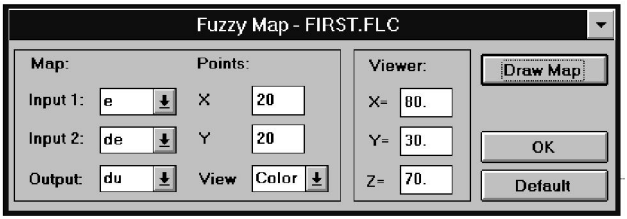
has no function at the moment.

4.10 Defuzzification



Choose the method of defuzzification, the variant and the limits of the output.
You have to enter the **limits of the output**. Whenever the values of the output exceed these value it will be limited to them.
Additionally you can make an **automatic integration** of the output and if you want, select **Defuzzification Variante for Simulation** (second is not implemented in this Version).
Available Defuzzification methods are Center of gravity (CoG), Maximum left (ML), Maximum right (MR) and Mean of Maximum (MoM). The variants for each defuzzification method are Singleton, Singleton & Area and Rectangle.

4.11 Fuzzy Map Dialog



(c) 1993-97 FDG-S.
N. Exler

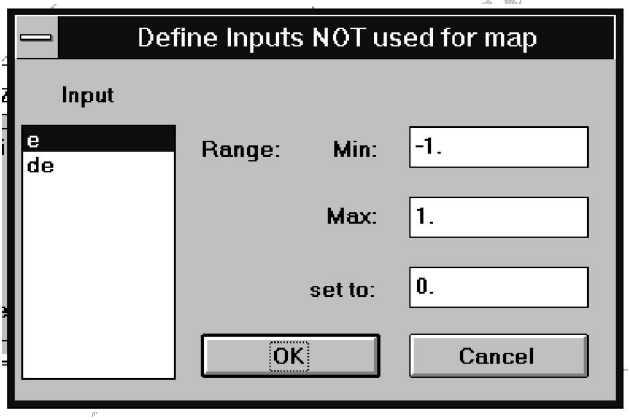
The Fuzzy Map dialog gives you the possibility of setting the input and output variables, the map resolution, the View type and the position of the viewpoint to the required values. On the left hand side you can select one or two variables for the inputs and one variable for the output in the combo boxes. The values of the input variables are automatically increased from the minimum up to the maximum value by the number of steps you have defined in the points edit boxes for the X- and Y-axis. You can choose between three different view types (Color, Net, White) for the fuzzy map, by selecting the appropriate type in the View combo box.

It is possible to change the coordinates of your viewpoint by changing one or more values in the View edit boxes (X, Y, Z).

By pressing the <***Default***> button the default values for all edit and combo boxes will be set. With the <***OK***> button you can delete the Fuzzy Map dialog and, if the Fuzzy Map View is already shown, it will also be deleted. When you press the <***Draw Map***> button the rules will be calculated for every point of the grid in the

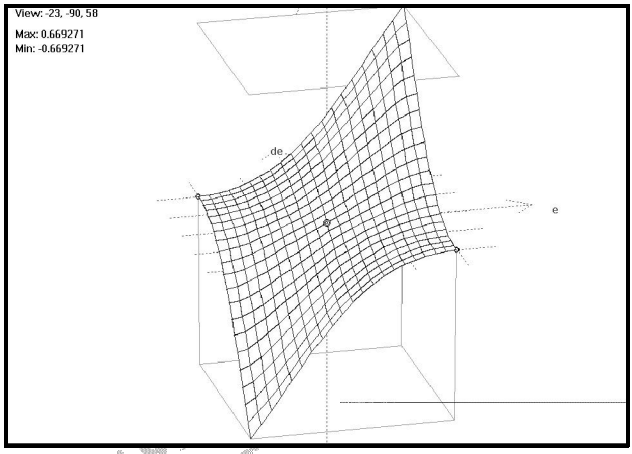
map. This can take a long time - so be patient!
After the 3-dimensional Fuzzy Map has appeared, yo u can look at it from any point you want.

4.12 Define Inputs NOT used for map



If you have more inputs defined in the fuzzy resourc e file than as you use for the map, the rest of the inputs can be freely defined. That dialog box shows a list of al l linguistic input variables. Select the one you want t o define as a constant input and enter the value. The range must be between the minimum and maximum valu e defined by the Fuzzy Resources. By default ever y linguistic variable is set to the mid value between th e minimum and maximum value.

4.13 Fuzzy Map View



This shows you a 3-dimensional fuzzy map of one output related to one or two input linguistic variables. After the Fuzzy Map has appeared, you can look at it from any point you want.

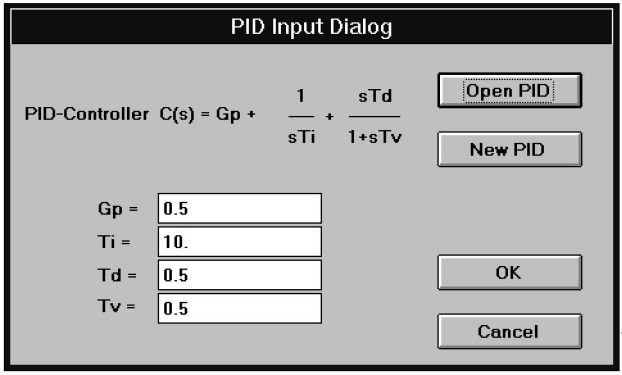
On the top left you can see the coordinates of your view point. You can **change the coordinates of your view point** in two different ways:

- You can go back to the Fuzzy Map Dialog and change the coordinates there or
- you can press the left mouse button, hold it and move the mouse. This will cause the moving of the fuzzy map and the respective changes in your view points coordinates. The left-right motion will be translated as a circular motion in the X-Y-plane of the two inputs; the up-down motion follows a sphere.

The minimum and maximum outp ut values shown on the map, and the coordinates of the view point are always in the top left corner of the window.
If you have chosen the **color view**, you will see on the left side under the Max-value a color bar with differen t colors corresponding to the different heights of the map.
If the **view is white** the grid of the cube in which th e map is placed will not be painted behind the map. If the **net view** is selected, between the calculated points jus t lines are drawn.
By double clicking the left mouse button the defaul t view will be restored.
By pressing the right mouse button you can walk ove r the map. The actual coordi nates will be shown on the top of the view, and marked in the map. A mousemovement to the right or left correspodents with the first linguisti c variable (X-axis), up-down movement correspodent s with the second linguistic variable (Y-axis).

(c) 1993-97 FDG-Systems N. Exler

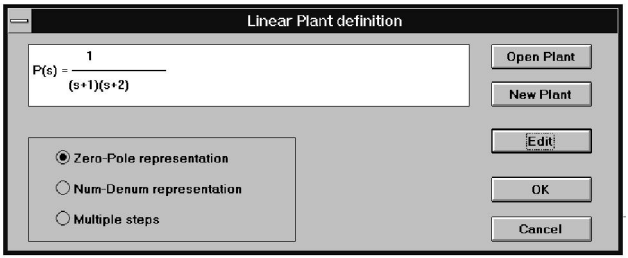
4.14 PID input dialog



Enter the factors of the standard **PID controller** you want to have here. The formula of calculation is written in the dialog box above the edit boxes. This controller is used for you to compare the results of a standard PI D controller to a **Fuzzy Controller**. You can do this by simulating the controller in a closed loop with a **plant** or just by studying the **step response** of the controller.

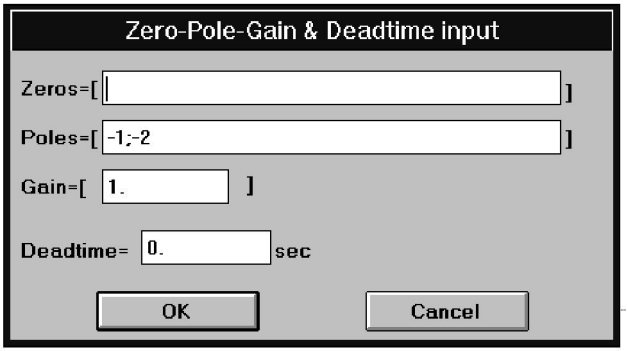
If you have deticated to solve the problem with a PID - controller, it is possible to convert it into an optimise d ADSP source code.

4.15 Linear Plant Definition - Zero-Pole representation



You need a plant to simulate the system in a closed-loop with the fuzzy controller to see the quality of your designed fuzzy controller.
By pressing the ***<Edit>*** button you can define a plant which is given by zeros, poles, gain and deadtime.
You also can design a classical PID-controller for a plant, but this is not the goal of FID Fuzzy Application Software.
The plant is shown in the list box in the top half of the Linear Plant Definition dialog box. If this list box is empty no plant is defined.
Each plant can be saved by pressing the ***<New Plant>*** button.
To open an already existing plant press the ***<Open Plant>***
button.

4.16 Zero-Pole-Gain & Deadtime input



The rules for entering the data are as follows:

- **Every zero** you have in the description of the plan t has to be entered as a numerical value. If you hav e more than one zero, then separate the different zeros with semicolons. If there is no zero in the plan t description leave the edit box clear.
- **Every pole** you have in the description of the plan t has to be entered as a numerical value. If you hav e more than one pole then separate the different poles with semicolons. If there is no pole in the plan t description leave the edit box clear.

Additionally you can define a gain of the plant and a deadtime. By default these two values are 1 and 0 respectively.

For example:

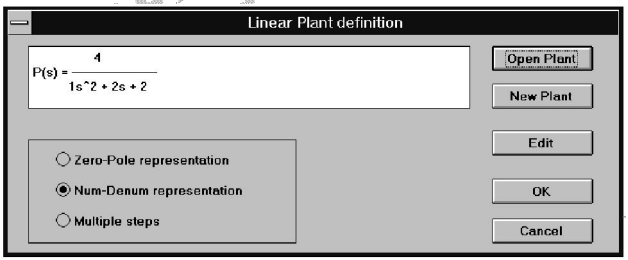
Zeros: -2;-1
Poles: -5;-4;-4
Gain: 1.5
Deadtime: 3 sec

will be interpreted as:

$$G(s) = 1.5 * \frac{(s+2)*(s+1)}{(s+5)*(s+4)*(s+4)} * e^{-3s}$$

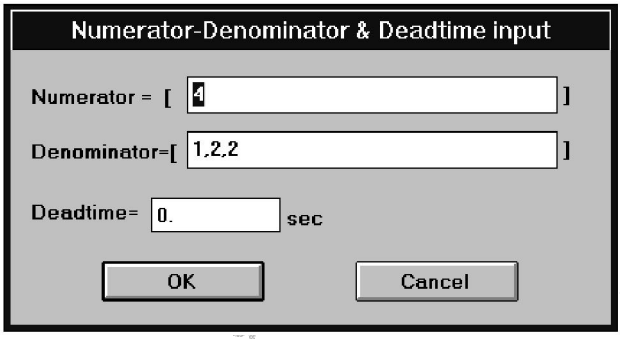
1993-97 FDG-Systems

4.17 Linear Plant Definition- Numerator-Denominator



This defines a plant given by the Numerator and Denominator. If you select the Num-Denom representation radio button and then press the ***<Edit>*** button you can define a plant by the numerator, denominator and deadtime. The numerator and denominator are polynomials in the Laplace space.

4.18 Numerator-Denominator & Deadtime input



You can enter the plant by entering the factors of th e polynomial representing the plant. Separate the factor s with a colon. The order of the polynomial is defined b y the numbers of factors entered in the edit boxes. So i f one factor is zero you have to enter a zero-value at th e right place, otherwise the plant is not correctly defined:

☞ Be sure that the order of the numerator is always less or equal to the order of the denominator.

Additionally you can define a dead-time of the plant.

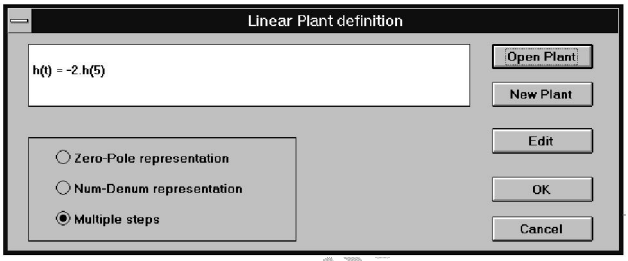
For example:

Numerator: 4,1
Denominator: 6,8,0,1

Will be interpreted as:

$$G(s) = \frac{4*s + 1}{6*s^3 + 8*s^2 + 0*s + 1}$$

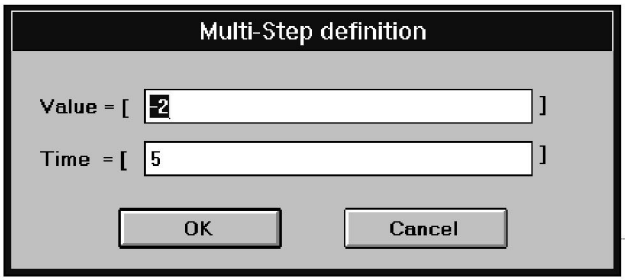
4.19 Linear Plant Definition - Multiple steps



If you select the Multiple steps radio button and the n press the **<*Edit*>** button, you can define a step o r multiple steps.
A step or multiple steps used for **simulating disturbance** is generally a kind of plant. This plant ha s no input but one output.

(c) 1993-97 FD
N. Exler

4.20 Multi-Step definition



Every step consists of two values, one in each edit bo x

and interpreted as one step.

☞ The step value and the step time have to be define d absolutely. You define at which time it will become which value. So the steps will not be added.

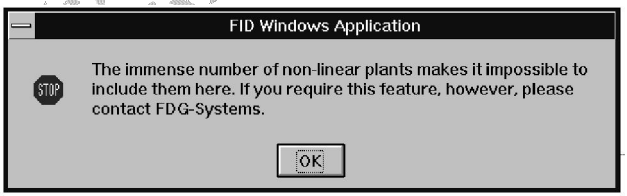
For example:

step value = -2
step time = 5 seconds

Will be interpreted as :
the step is zero before 5 seconds and afterwards -2.

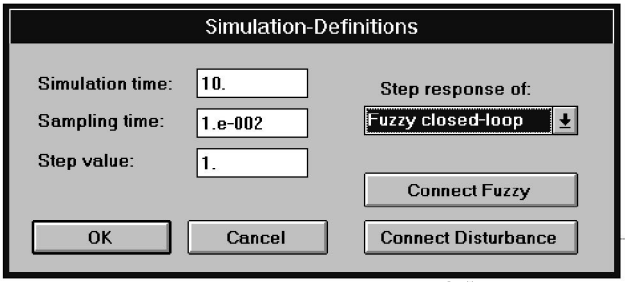
(c) 1993-97 FDG-Systems

4.21 Non-linear Plant Definition


FID Windows Application
The immense number of non-linear plants makes it impossible to include them here. If you require this feature, however, please contact FDG-Systems.
OK

Unfortunately there are very many different non-linea r plants in the real world and everyone needs a differen t one. So if you really need it for simulating a specia l problem, feel free to contact us for a consultation. Please describe the problem very accurately by mathematica l means or measured values and consider if you reall y cannot linearise it.

4.22 Simulation Definition



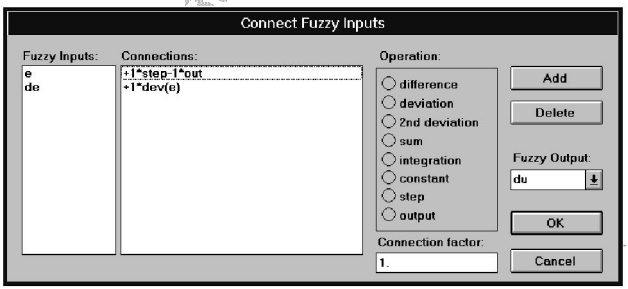
The Simulation Definition dialog box above shows you the basic settings for all simulations.
In the three edit boxes you can change the simulation time, the sampling time and the step value. The simulation time always starts at zero and ends at the time which is written in the edit box "Simulation time". With the Sampling time you can choose the discretion in time of the simulation. i.e.: This is the time between two simulated values.

For example:
If you have a simulation time of 10 seconds with a sampling time of 0.01 seconds this will need a total of 1000 calculation steps. If you have a rulebase with nine rules then 9000 rules must be calculated for the simulation time. Depending on your computer system, this might take a bit longer!

The step value is the reference value for closed-loop systems, otherwise it is the input value in a plant or PID-controller, or one input in a fuzzy module.
In the combo box "Step response of" you can choose the response of a plant, PID-controller, Fuzzy module, PID

closed-loop system or a Fuzzy closed-loop system.
If you choose the response of a plant or PID-controlle r no other settings are possible for the simulation. If yo u choose Fuzzy or Fuzzy closed-loop you must decid e how to connect all inputs to the fuzzy module. To do this press the <***Connect Fuzzy***> button. If you want to simulate the influence of a disturbance on a closed-loop control, press the <***Connect Disturbance***> button (only available for PID closed-loop and Fuzzy closed-loop).

4.23 Connect Fuzzy Inputs



In this dialog box you can define all connections fro m the inputs and outputs at the fuzzy controller.
The first list box is only a list of all linguistic inpu t variables defined in the fuz zy resource file. In the second list box you can add t he wanted connections by selecting an operation and a connection factor and then pressin g the <***Add***> button. If you make a mistake you can delete the whole connection list for each input by selecting the wrong connection list and pressing the <***Delete***> button.

The output of the fuzzy module which should be used as the input to the plant or just as the calculated output, can be selected in the combo box above the **<*OK*>** button. The connection list has two different interpretation s depending on the selection of the step response in th e Simulation Definition dialog.
If **Fuzzy** is selected the connection list is interpreted a s follows (example from the dialog above):

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & -1.0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} h_{k+1} - h_k \\ (h_{k+1} - h_k)/T_s \\ (h_{k+2} - 2 * h_{k+1} + h_k) \\ \Sigma h_k \\ T_s * \Sigma h_k \\ 1.0 \\ h_k \\ y = du \end{bmatrix}$$

(c) 1993-97 FDG-Systems N. Exler

h .. step value
y .. output value (du)

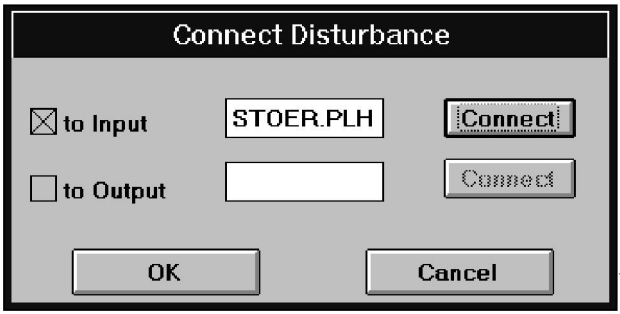
If **Fuzzy closed-loop** is selected the connection list i s interpreted as follows (example from the dialog bo x above):

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & -1.0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} e_{k+1} - e_k \\ (e_{k+1} - e_k)/T_s \\ (e_{k+2} - 2 * e_{k+1} + e_k) \\ \Sigma e_k \\ T_s * \Sigma e_k \\ 1.0 \\ h \\ y \end{bmatrix}$$

(c) 1993-97 FDG-Systems N. Exler

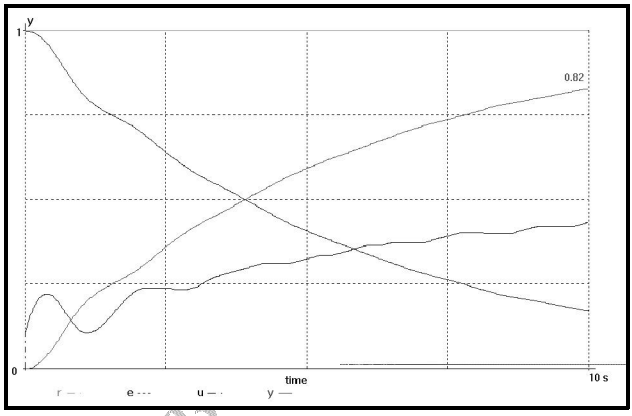
h .. reference value (step value)
y .. output value (output value of the plant)
e .. error value e = h - y

4.24 Connect Disturbance

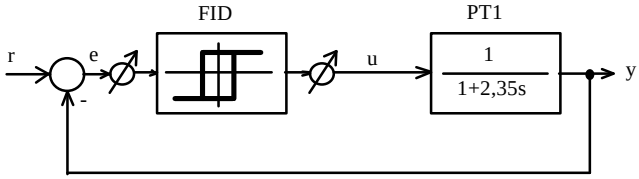


You can either connect a step or multiple steps to th e input and/or to the output of the closed loop system.
To connect, check the check box for the input and/o r output and press the appropriate <***Connect***> button and select a plant by means of the plant definition (Multiple steps with the extension PLH).
To eradicate the distur bance to the input or output cancel the according check box.

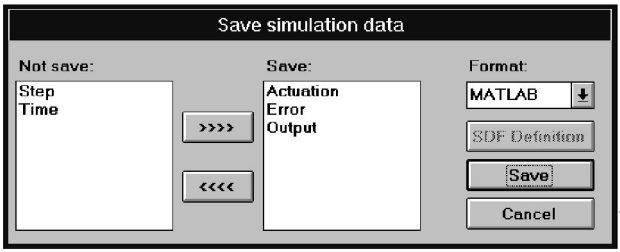
4.25 Simulation View



This window shows the result of a defined simulation . By default the output of the controller, plant or close d loop system will be shown. By clicking a button on th e tool bar, the view will toggle the visibility of the input , error, output or the actuation. The channels correspon d to the following diagram:



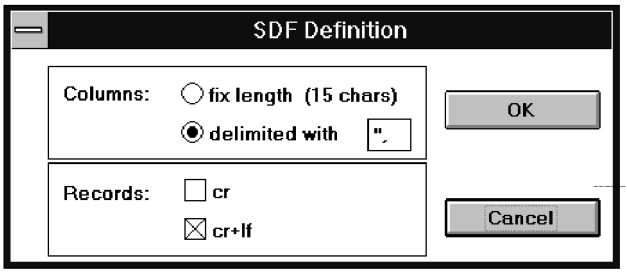
4.26 Save simulation data



After you have simulated a system you can store, before you close the simulation view, the simulated data in a file. The format can be selected from the combo box Format. You can choose between two data formats, MatLab or a Standard Data Format. If Standard Data Format is selected then it is possible to press the <***SDF Definition***> button for further settings.

There are two list boxes with all available data. Select the data you want to store by selecting them in the "not save" list box and moving them into the "Save" list box by pressing <">>>>"> button.
After pressing the <***Save***> button you will be asked for the filename and after entering a valid path and name the data will be saved in the selected format.

4.27 SDF Definition



(c) 1993-97 FDG N. Exler

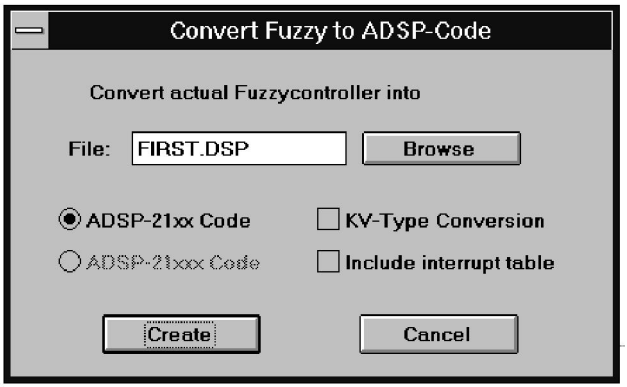
Standard Data Formats can have different formats.

- Either the length of each field has a particular fixe d length no matter of the data size in that field or
- the data are separated by delimiters. Each record i s terminated with a carriage return (cr) or a carriag e return + line feed (cr+lf).

In the FID program both versions are available. B y default a fixed length with 14 chars per field will b e taken. If you choose ***<delimited with>*** the first character in the edit box will be used at the beginning and the end of each field, the second character separating th e different fields.

The data will be stored in the order displayed in the save list box of the save simulation data dialog box.

4.28 Convert Fuzzy to ADSP-Code



Whenever you want to create an ADSP source code choose the menu item "ADSP Source code". Enter the name and path of the source code file by pressing the <***Browse***> button and for what ADSP you want to generate the source code (not implemented in this version).

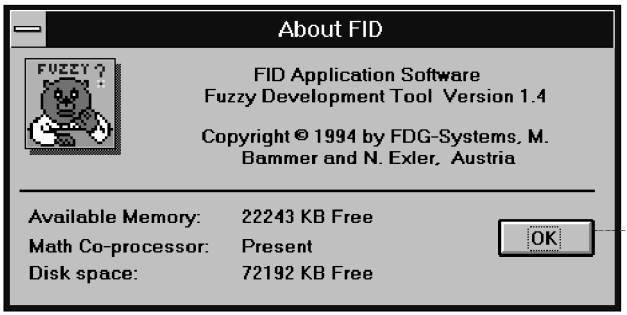
Check the **KV-Type Conversion** box, if you only have a Karnaugh-Veitch - Rulebase. The resulting code will be faster, shorter and more efficient than the expert type rulebase conversion. For more details see the documentation of the source code.

If the **Include interrupt table** check box is checked than a default interrupt table will be included in the source code. The default file name for the interrupt table is "**inthand.dsp**".

The radio button for the floating point ADSP-21xx x code generation is grayed and disabled. Only ADSP -

21xx code generation is possi ble in this software version.

4.29 About FID


About FID
FUZZY ?
FID Application Software
Fuzzy Development Tool Version 1.4
Copyright © 1994 by FDG-Systems, M. Bammer and N. Exler, Austria
Available Memory: 22243 KB Free
Math Co-processor: Present
Disk space: 72192 KB Free
OK

Shows you important information about the FID - program and the computer.

(c) 19?? N. Exler

5. Fuzzy-Logic on an ADSP-21xx

5.1 Features

- Up to 12-bit high-resolution for input variables and membership values and up to 16-bit high resolution for output variables.
- Fuzzy logic operation in sequential-mode or loop-mode.
- In loop-mode the number of input variables is restricted to a maximum of 3, but with a hierarchical structure the number of input variables is unlimited (memory limit!)
- In loop-mode very little memory space is necessary => low cost version ADSP-2105/06 can be used => especially for fuzzy control applications.
- In sequential-mode a large number of antecedents and consequents can be used => especially for fuzzy control and fuzzy expert systems.
- High-speed inference processing by single cycle instructions.
  24,9 µs for 2 conditions (antecedents) and 1 consequent, 49 rules and 16 MHz.
- Efficient fuzzification and defuzzification for any number of input and output variables.
  15,5 µs for fuzzification with 2 input values with 7 membership functions each.
  3.84 µs for a crisp value evaluation of 7 output MFs of the second or third type of center-of-gravity method.

(c) 1993-97 FDG-Systems N. Exler

- Up to 1kx16 Bit on-chip data RAM.
- Up to 2kx24 Bit on-chip programm RAM, enoug h for almost all control problems.
- Interfaces to standard microprocessors.
- Modified Harvard architecture.
- No other processors are necessary for mixed control strategies (classical (P, PI, PD, PID, state space) and fuzzy).
- Easily readable algebraic assembler code.

(c) 1993-97 FDG-Systems N. Exler

5.2 Function List

Fuzzification:

Any number of input variables are allowed
Triangular and trapezoid shapes for the membership functions

Inference Functions:

Loop-mode:

Indirect-addressing of antecedents and consequents
For 2 antecedents and 1 consequent the following methods are used:
MAX-MIN Inference
MAX-ALG-PRO Inference
ALG-SUM-MIN Inference
ALG-SUM-PRO Inference
For 3 antecedents and 1 consequent the following methods are used:
MAX-MIN Inference

Sequential-mode:

Direct addressing of antecedents and consequents
No restriction on the number of inputs
MIN, MAX, ALG-SUM, ALG-PRO, BOU-SUM, BOU-PRO operators for the inference

Defuzzification:

Any number of membership values for each output
Three different kinds of center-of-gravity methods
- consequent MFs are represented as singletons
- consequent MFs are represented as singleton s and areas
- consequent MFs are represented as rectangula r functions

Tuning by scaling factors:

Scaling factors for all input and output variable s are individual

(c) 1993-97 FDG-Systems N. Exler

5.3 Specifications

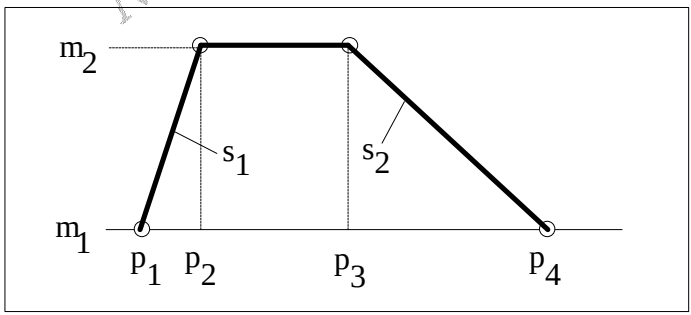
5.3.1 Fuzzification

First of all, the input values must be fuzzified in order to assign the input values by certain grades to the fuzz y sets. The fuzzification interface is therefore the bridg e between the measured or evaluated crisp values and the compositional rules of inference for fuzzy logi c evaluation. The fuz zy sets are restricted to triangular and trapezoid shaped membership functions (MFs). Th e fuzzification process determines for every input valu e

the membership values (MVs) of all fuzzy sets belonging to a linguistic variable. Then the fuzzification interface stores all membership values of an input variable in a field of variables (linguistic labels). The resolution of the input variables can be any number of bits up to 12 bits, and the membership value resolution is always 12 bits for the MFs mentioned above.
The general shape of a membership function is the trapezium and it is characterized by 8 values (see fig. 1). A membership function can be defined for each fuzzy set (linguistic label).

(c) 1993-97 FDG-Systems N. Exler

There are some constraints for the definition of MFs:
- The vertices should be connected by straight lines.
- The slope evaluation should be made by truncating the resulting decimal value.
- The slope at a right angle must be 4000h.



init order: p1 , m1 , p2 , s1 , p3 , m2 , p4 , s2

Fig. 1: General shape of membership functions

The trapezium can, by certain definitions, take on the shape of a triangle or rectangle.

The following (figure 2) shows examples of MFs tha t can be so defined.

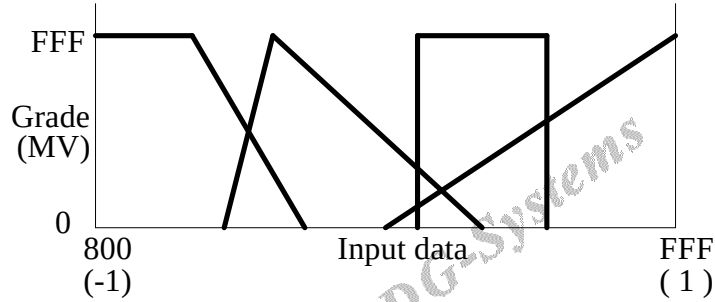


Fig. 2: Possible membership functions which are allowed

(c) 1993-97 FDG-Systems N. Exler

The fuzzification process is a process to obtai n membership values (grades) for a linguistic variable b y using the input value and all membership functions fo r this input (see fig. 3).

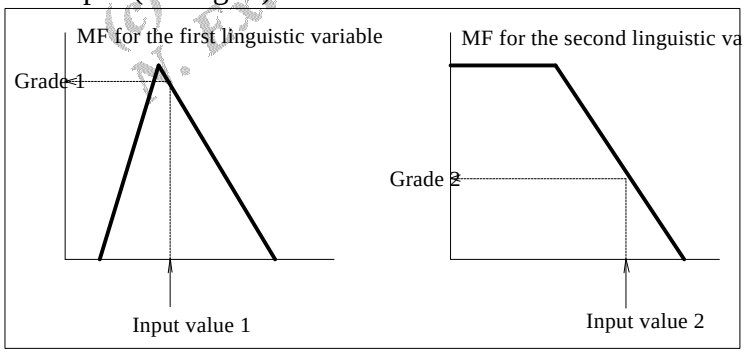


Fig. 3: The fuzzification process in principle

Memory Model for the fuzzification interface:

A simple example for the memory model of a FID - controller (FID - Fuzzification-Inference - Defuzzification) with two input variables and two membership functions each is shown in fig. 4. The necessary memory space for the input variables and the membership values must be arranged in data memory locations. The necessary data for indirect addressing and for the MFs is stored in program memory locations.
The indirect addressing capability makes the necessary program code very short and the result is a very fast and accurate fuzzification interface.

(c) 1993-97 FDG-Systems N. Exler

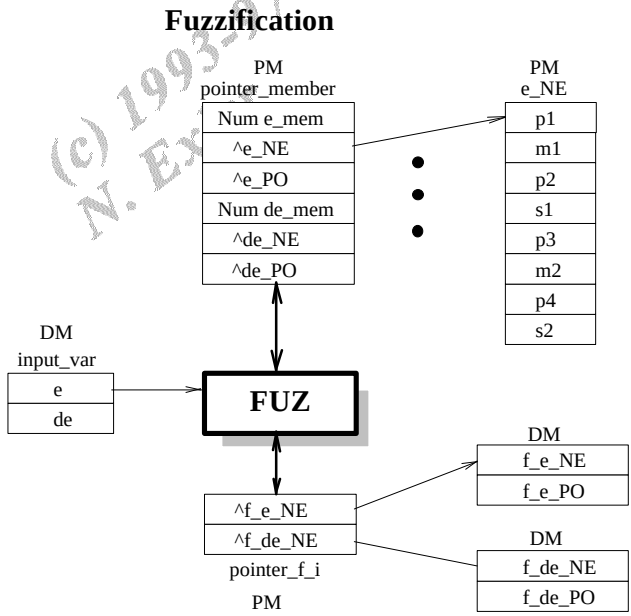


Fig. 4: Memory model for the fuzzification interface

The fuzzification process needs:

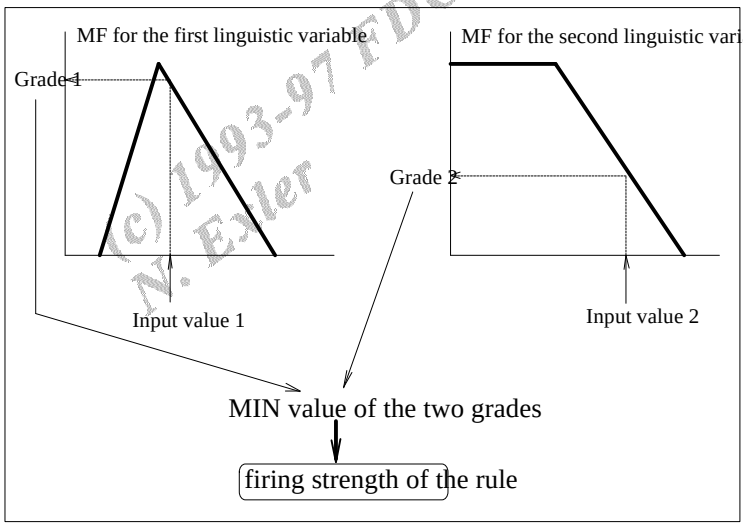
- a field for all input variables (linguistic variables) in data memory (e.g. input_var in DM) and an I-register which points to the beginning of this field.
- for every membership function (triangular or trapezoid) 8 program memory locations.
- for every input variable (linguistic variable) any number from one to seven membership functions (fuzzy sets) i.e. 8 to 56 program memory locations ; more than seven membership functions is mostly not useful or necessary
- a field with the number and start addresses of all the membership functions for all input variables, and an I-register to point to the first address in this field for indirect addressing.
- for every input variable a field for the membership values with as many membership functions.
- a field for the start addresses for all membership value fields and an I-register to point to the first address in this field.

(c) 1993-97 FDG-Systems N. Exler

5.3.2 Inference mechanism

The whole inference mechanism is based on Zadeh's compositional rule of inference and comprises two major tasks. The first one is the evaluation of the premise part (IF part) of the production rules, and the second task is the union of all conclusions (THEN parts). The premise comprises two or more antecedents which are normally evaluated by means of triangular norms (e.g. MIN, ALG-

PRO, BOU-PRO, DRA-PRO).
Antecedent processing is the process of obtaining th e firing strength of a production rule using the grades o f different input variables. If Mamdani's mini-fuzz y implication is used, the m inimum operator is used for the antecedent processing. The firing strength of the rule i s the minimum of all antecedents (see fig. 5). Thi s minimum value is used in consequent processing (union of all conclusions).


MF for the first linguistic variable
MF for the second linguistic vari
Grade 1
Grade 2
Input value 1
Input value 2
MIN value of the two grades
firing strength of the rule
(c) 1993-97 FDG-Systems N. Exler
Fig. 5: Antecedent processing with two grades

In consequent processing, the firing strength of the rul e obtained in antecedent processing is compared with th e firing strength of other rules so that a fuzzy output i s obtained for each consequent label. In Mamdani's Max-Min compositional rule of inference method the fuzz y

output is evaluated by the maximum operator (see fig. 6).

In loop-mode the two tasks are not separated. They are combined via the memory model and indirect addressing facility to one iterative task. In sequential-mode it is possible to separate the two tasks or to perform the same combined processing strategy as in the loop-mode.

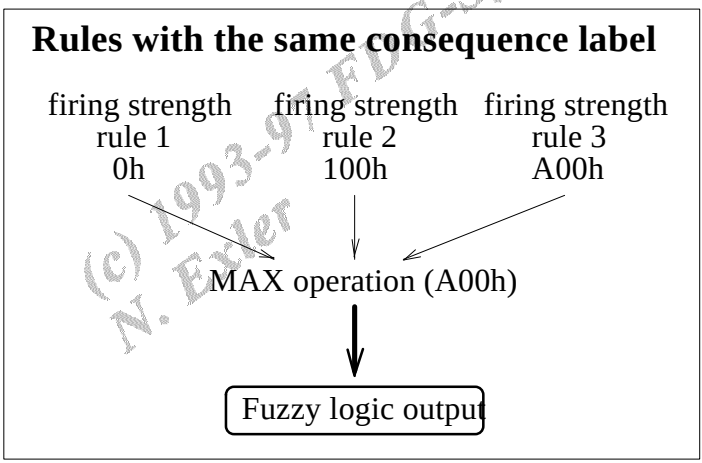

Rules with the same consequence label
firing strength rule 1 0h
firing strength rule 2 100h
firing strength rule 3 A00h
MAX operation (A00h)
Fuzzy logic output
(c) 1993-97 FDG-Systems N. Exler

Fig. 6: Consequent processing for three values with the same consequent label.

A simple example with three rules out of a set of rules should clarify the two possibilities,

separate antecedent and consequent processing:

IF (e is PO) AND (de is NE) THEN (du is PO)
IF (e is PO) AND (de is PO) THEN (du is PO)
IF (e is PO) AND (de is ZE) THEN (du is PO)

combined antecedent and consequent processing:

```
IF (e is PO)  AND  (de is NE)     OR
   (e is PO)  AND  (de is PO)     OR
   (e is PO)  AND  (de is ZE)     THEN (du is PO)
```

As mentioned above there are two modes for the production rule processing. The modes are the loop-mode and the sequential-mode. The loop-mode is designed for fuzzy control applications with two or three antecedents (inputs). It is possible to build a cascade (hierarchical structure) of two or three input FID-modules because it is possible to use the fuzzy output of the first FID-module as an input for the second FID-module (see fig. 7). The resulting program code is very short and the inference speed is very high.

(c) 1993-97 FDG-Systems N. Exler

Fuzzy-PID controller with a cascade structure

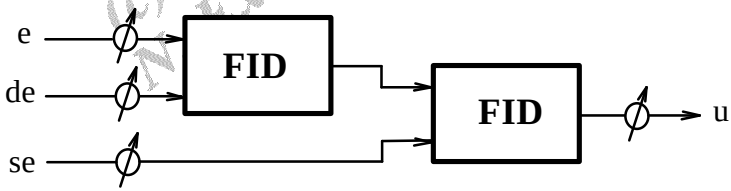


Fig. 7: Hierarchical structure with the same FID-modules

The sequential-mode is also designed for fuzzy control applications as well as fuzzy expert systems, but with no restrictions on the number of antecedents and consequents. The only restriction is the available memory space of the ADSP-21xx. In sequential-mode it is possible to have fewer or more production rules than the product of the number of MFs.

Memory model for the inference mechanism in loop-mode

The same example as for the fuzzification memory model is used. The membership values (grades) for each input variable are stored in separate data memory fields. The output membership values (fuzzy logic output) are also stored in a data memory field. The necessary rulebase for indirect addressing is located in program memory and needs one memory location for each IF-THEN-rule. In loop-mode an intensive use of the I- and L-registers is made to get a very short and fast inference processing (see fig. 8).

(c) 1993-97 FDG-Systems N. Exler

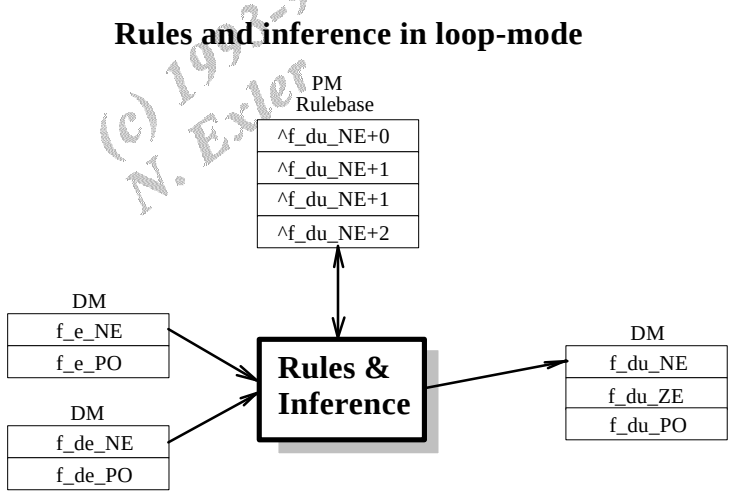


Fig. 8: Memory model for the inference mechanism in loop-mode

The inference process needs:

- a data memory field for each input membership value field (circular buffers) and an I- and L- register

to point to these fields (1,2 or 3 I's and L's).
- a program memory field for all rules and an I-register to point to the rulebase.
- a data memory field for the output membershi p values and an I-register to point to this field.

Memory model for the inference mechanism i n sequential-mode

The membership values or antecedents for each inpu t variable are stored in separate data memory fields. Th e output membership values are also stored in a dat a memory field. For direct addressing it must be possible to reference each value in a data field by its name (se e fig. 9). This facility makes it possible to use as man y antecedents as desired and to use different triangula r norms for antecedent processing.

Instead of the rule base in a program memory field, th e rulebase is coded as individual instructions in progra m memory. The inference mechanism in sequential-mod e needs, therefore, much more program code than in loop-mode, but it is very flexible in that norms for antecedent and consequent processing can be selected separately . There is no restriction on the number of antecedents and consequents.

(c) 1993-97 FDG-Systems N. Exler

Rules and inference in sequential-mode

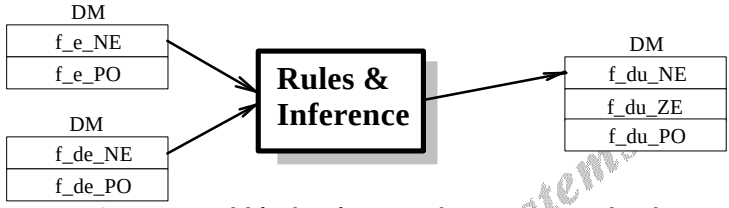


Fig. 9: Memory model for the inference mechanism in sequential mode.

The inference process needs:

- data memory fields for the MVs of each input variable (circular buffers are not necessary).
- data memory fields for the MVs of each output variable (circular buffers).
- names for each membership value for direct addressing.

(c) 1993-97 FDG-Systems N. Exler

5.3.3 Defuzzification

In order to get a crisp output value it is necessary to defuzzify the fuzzy logic output data. Three different kinds of the center-of-gravity method are implemented for defuzzification.

The first type uses just singletons for the output MFs which represent the center of the output MFs. For an accurate defuzzification, from the mathematical point of view, it is necessary to use symmetrical MFs of the same size. The arrangement of these MFs should be regular along the abscissa (linguistic variable).

The evaluation used for defuzzification is shown below:

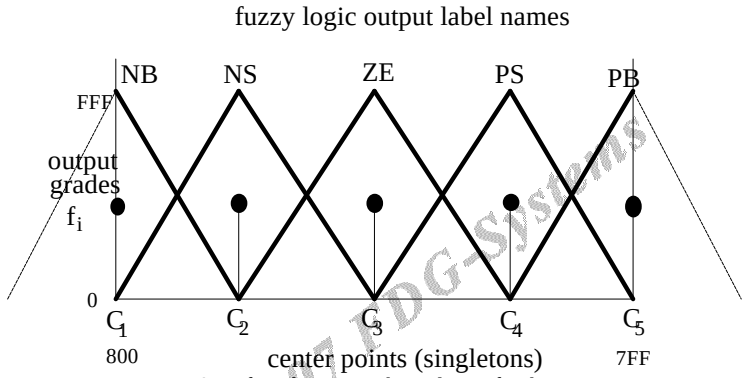


Fig. 10: Defuzzification with singletons for the MFs.

The crisp value for the output can be calculated by using the following formula:

$$crisp\ value = \sum (f_i * C_i) / \sum f_i$$

The second type uses singletons and scaled areas for the output MFs. The singletons represent the center of the output MFs and the areas are scaled to the largest area of the MFs. The only restriction is that the MFs should be symmetrical.

The evaluation used for defuzzification is shown below:

(c) 1993-97 FDG-Systems N. Exler

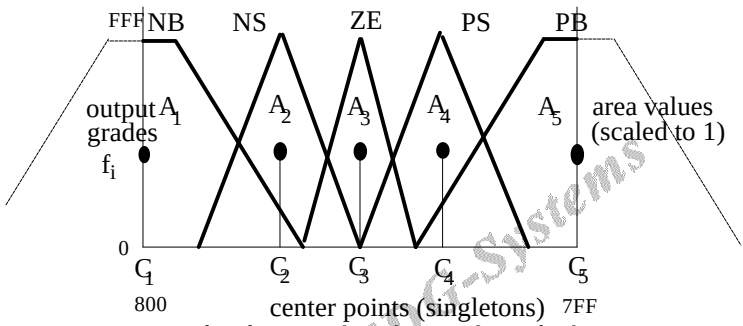


Fig. 11: Defuzzification with singletons and areas for the MFs.

(c) 1993-97 FDG-Systems N. Exler

The crisp value for the output can be calculated by using the following formula:

$$crisp\ value = \sum (f_i * A_i * C_i) / \sum f_i * A_i$$

e.g. IF a1 and a5 are equal and bigger than a2 , a3 , a 4 then the scaled areas are:

A1 and A5 set to 1 (1.15) and A2 = a2 / a1,
A3 = a3 / a1, A4 = a4 / a1

The third type uses rectangular MFs for the output fuzzy sets. These MFs are characterized by their points at th e abscissa. In order to use the same defuzzificatio n method, as for the second type, it is necessary to stor e instead of areas, the distances between the points next to each other. For the centers it is necessary to store th e mean value of two adjoining points.

The evaluation used for defuzzification is shown below:

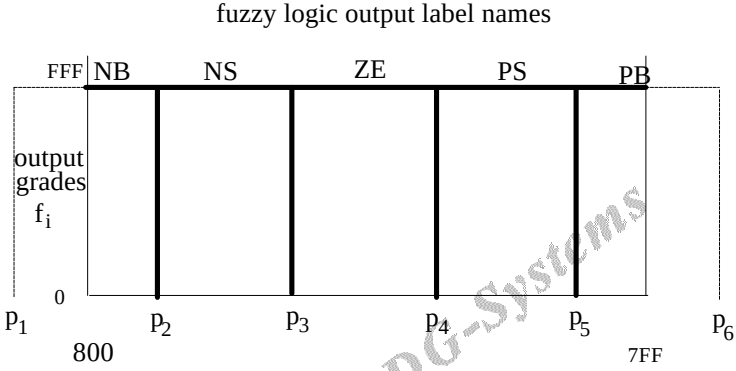


Fig. 12: Defuzzification with rectangular MFs.

(c) 1993-97 FDG-Systems N. Exler

The crisp value for the output can be calculated by using the following formula:

$$crisp\ value = \sum (f_i * (p_{i+1} - p_i) * (p_{i+1} + p_i) / \sum 2 * f_i * (p_{i+1} - p_i)$$

When the denominator = 0, division does not take place and the crisp value is set to 000h. This is the valu e between 800h and 7FFh.

Memory model for the defuzzification

The same example as for the fuzzification and th e inference memory model should be used. The outpu t membership values (fuzzy logic output) are stored in a data memory field. The necessary values (centers, areas, distances or mean values) for the defuzzification ar e stored in a program memory field. The length of thi s field depends on the type of center-of-gravity method. A variable in data memory for the crisp output value i s

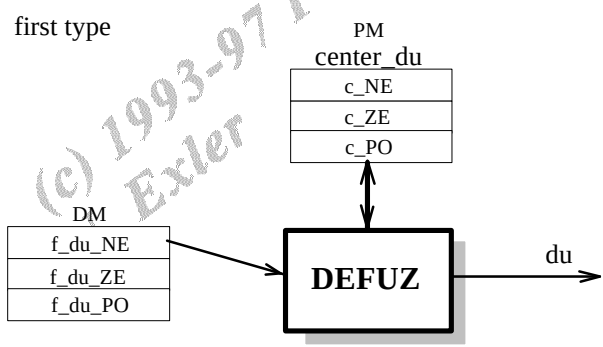
needed.

The defuzzification needs:

- an I-register to point to the fuzzy output field and the corresponding L-register to store the length of this field.
- an I-register to point at the singleton field, singleton and area field or the distance and mean value field.
- a DM-variable to store the crisp value.

(c) 1993-97 FDG-Systems Exler

Defuzzification



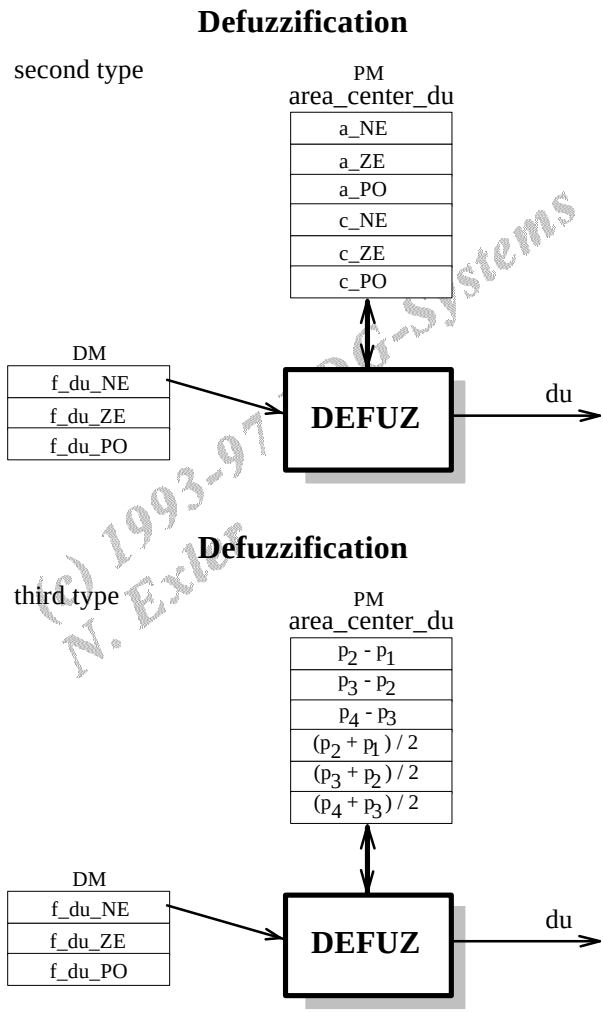


Fig. 15: Memory models for all kinds of defuzzification.

5.3.4 Tuning

An easy to use tuning facility for the input and outpu t variables are the scaling factors. Their role is to tune the fuzzy controller to obtain the desired dynamic properties of the process-control loop. So the input and output ar e changed proportionally by these scaling factors. The y can also be used for a fine and coarse switching between different dynamic behaviour by using the same FI D elements.

(c) 1993-97 FDG-Systems N. Exler

Input variable tuning

The scaling factors ar e stored in a program memory field with as many locations as the input variable field in data memory. The range of the scaling values extend from 0 to 16. The resulting changed input values are stored i n the same input variable fiel d, but the format of the values is changed from a signed 12 bit to an unsigned 12-bi t format.

The memory model is shown below:

Input-variable tuning

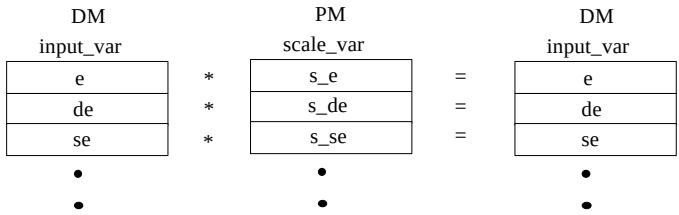


Fig. 16: Memory model for input variable tuning.

The input variable tuning needs:

- an I-register to point to the data memory field wit h the input values as the corresponding L-registe r stores the number of input values.
- an I-register to point to the program memory fiel d with the scaling values.

Output variable tuning

The scaling factors for the output variables are store d separately in program memory locations.

(c) 1993-97 FDG-Systems N. Exler

The output variable tuning needs:

- the crisp output values after the defuzzification i n data memory locations (single precision) for direc t addressing.
- the scaling values in program memory locations fo r direct addressing.
- two data memory locations for storing the change d crisp output values (double precision).

5.3.5 Resolutions and Formats

Input values and scaling factors

The input values can have a re solution of up to 12 bits. A signed 12-bit format is used for input values. i.e. 800h is equal to -1 (5.11) and 7FFh is equal to +1-1 LSB (5.11). The input variable tuning changes the input values b y multiplying them by the scaling values.

The resolution of the scaling factors is 16-bits and th e

chosen format is signed 5.11 with values from -16 to 16.

Membership functions for the input values

The p_i - values are in a signed 12-bit format and the membership values m_i are in an unsigned 12-bit format. The values for the slopes s_i are in a 12.4 format. Therefore the slope values can extend from 1/16 (theoretically) or better 1 (practically; see possible MFs) to 4096.

Membership values (grades)

All membership values are in an unsigned 12-bit format (0 to FFFh).

Membership functions (singletons) for the fuzzy output

The c_i - values are in a signed 12-bit format. The scaled areas are in an unsigned (1.15) format. The distances are in a signed 1.15 format and the mean values are in a signed 12-bit format.

Crisp output values

The crisp output values use a signed 12-bit format and they must be changed, if necessary, to an unsigned 12-bit format.

(c) 1993-97 FDG-Systems N. Exler

5.3.6 Implementation of the generated ADSP-21xx source code in your specific application

This example (FPID.DSP) explains the implementation of the fuzzy ADSP-21xx source code to create a running sample on the EZ-LAB board or on your specific target hardware. Additional code segments must be implemented at the following positions, marked by extra lines, to make a running example on the EZ-LAB board.

Change the boot-page, the absolute start address and/or the module name, if necessary, for your application.

```
.MODULE/BOOT=0/ABS=0                                FPID_FLC;
{*******************************************}
{***                                         }
{***       ADSP Sourcecode-Generation        }
{***                                         }
{***  (c) 1993-1994 Bammer, Exler            }
{***      FDG - Systems  (mb/en)             }
{***  Created: Thu Dec 08 17:04:59 1994      }
{***  Fuzzy-sourcefile: FPID.FLC             }
{***  generated for: ADSP 21xx  Serialized   }
{***                                         }
{*******************************************}


#include <fuztools.lib>
```

If necessary include other libraries here.

```
{ External calls }
.EXTERNAL   scaling_input;
.EXTERNAL   FUZZIFY;
.EXTERNAL   DEFUZ_WC;

{ Declaration of fuzzy-variables }
{ error:  e  Input }
.VAR/PM/CIRC     e_neg[8];
.VAR/PM/CIRC     e_ze[8];
.VAR/PM/CIRC     e_pos[8];
.VAR/DM/CIRC     f_e[3];
{ sum error:  se  Input }
.VAR/PM/CIRC     se_neg[8];
```

(c) 1993-97 FDG-Systems N. Exler

```
.VAR/PM/CIRC     se_ze[8];
.VAR/PM/CIRC     se_pos[8];
.VAR/DM/CIRC     f_se[3];
{ diff error:  de  Input }
.VAR/PM/CIRC     de_neg[8];
.VAR/PM/CIRC     de_ze[8];
.VAR/PM/CIRC     de_pos[8];
.VAR/DM/CIRC     f_de[3];
{ actuation:  u  Output }
.VAR/DM/CIRC     f_u[3];
.VAR/PM/CIRC     center_u[3];
.VAR/DM          scale_u;
.VAR/DM          u_HSW, u_LSW;

{ further declarations: }
.VAR/PM/CIRC     pointer_member[12];
.VAR/DM/CIRC     input_var[3];
.VAR/PM/CIRC     scale_var[3];
.VAR/PM/CIRC     pointer_f[3];
```

Include non-fuzzy variables and port label declarations here

(c) 1993-97 FDG-Systems N. Exler

```
{ Initialization: p1,m1,p2,s1,p3,m2,p4,s2}
.INIT     e_neg:H#F80000, H#0FFF00, H#F80000, H#400000, H#F80000,
H#0FFF00, H#000000, H#001000;
.INIT     e_ze:H#F80000, H#000000, H#000000, H#001000, H#000000,
H#0FFF00, H#080000, H#001000;
.INIT     e_pos:H#000000, H#000000, H#080000, H#001000, H#080000,
H#0FFF00, H#080000, H#400000;
.INIT     se_neg:H#F80000, H#0FFF00, H#F80000, H#400000, H#F80000,
H#0FFF00, H#000000, H#001000;
.INIT     se_ze: H#F80000, H#000000, H#000000, H#001000, H#000000,
H#0FFF00, H#080000, H#001000;
.INIT     se_pos: H#000000, H#000000, H#080000, H#001000, H#080000,
H#0FFF00, H#080000, H#400000;
.INIT     de_neg: H#F80000, H#0FFF00, H#F80000, H#400000, H#F80000,
H#0FFF00, H#000000, H#001000;
.INIT     de_ze: H#F80000, H#000000, H#000000, H#001000, H#000000,
H#0FFF00, H#080000, H#001000;
.INIT     de_pos: H#000000, H#000000, H#080000, H#001000, H#080000,
H#0FFF00, H#080000, H#400000;

.INIT     center_u:   H#F80000,H#000000,H#080000;
.INIT     scale_var:    H#019900,H#001400,H#080000;
.INIT     scale_u:   H#2000;
```

Include further variable initializations here.

```
{---------------------------------------------}
{----------  Start of ADSP-program  -----------}
{---------------------------------------------}
```

Include the interrupt table here

```
{!!!
```

```
     AR = 3;
     PM(I4,M5) = AR;
     AR = ^de_neg;
     PM(I4,M5) = AR;
     AR = ^de_ze;
     PM(I4,M5) = AR;
     AR = ^de_pos;
     PM(I4,M5) = AR;

     I4 = ^pointer_f;
     L4 = %pointer_f;
     AR = ^f_e;
     PM(I4,M5) = AR;
     AR = ^f_se;
     PM(I4,M5) = AR;
     AR = ^f_de;
     PM(I4,M5) = AR;

{ Initializing linguistic Variables and Rulebase }

     I0 = ^f_e;
     L0 = %f_e;
     init_member(0x0000,I0,L0);

     I0 = ^f_se;
     L0 = %f_se;
     init_member(0x0000,I0,L0);

     I0 = ^f_de;
     L0 = %f_de;
     init_member(0x0000,I0,L0);

     I0 = ^f_u;
     L0 = %f_u;
     init_member(0x0000,I0,L0);
```

(c) 1993-97 FDG-Systems N. Exler

Include interrupt settings like ICNTL, IFC, IMASK here and enable timer interrupt if necessary. The IDLE loop should also be included here.
The interrupt service routines start here.

```
{-------------------------------------------------}
{ The following lines are the real control code.  }
{ If you want to make a loop, it is best to start }
{ above this comment!                             }
```

The following should be part of an interrupt service routine from here onwards. e.g. in a timer-interrupt

service routine.

```
     I0 = ^input_var;
     L0 = %input_var;
     I4 = ^scale_var;
     L4 = %scale_var;

{ You can NOW make any necessary calculations for the inputs.}
{ Store each value according to the initializing order       }
{ simply with                                                }
{                DM(I0,M1)=AR                                }

  Store the INPUTS here
```

Include all necessary inputs in the right order here. The right order can be seen from the initializing procedure of fuzzy sets above. The input values can come from an ADC, a serial port and/or a memory mapped host. Make sure you calculate the input values correctly so as to run the ADSP source code like the simulation in the FID Application Software.

(c) 1993-97 FDG-Systems N. Exler

```
{ scaling the inputs }
     call scaling_input;

{ Fuzzification }
     I4 = ^pointer_member;
     L4 = %pointer_member;
     I5 = ^pointer_f;
     L5 = %pointer_f;
     call FUZZIFY;

{ Inference: Serialized Type }
Rule1:
     AR = DM(f_e+H#02 );
     AY0 = DM(f_se+H#02 );
     MAX_OP;
     AY0 = DM(f_de+H#02 );
     MAX_OP;
     DM(f_u+H#02 ) = AR;
Rule2:
     AR = DM(f_e+H#01 );
     AY0 = DM(f_se+H#01 );
     MAX_OP;
     AY0 = DM(f_de+H#01 );
     MAX_OP;
     DM(f_u+H#01 ) = AR;
```

```
Rule3:
     AR = DM(f_e+H#00 );
     AY0 = DM(f_se+H#00 );
     MAX_OP;
     AY0 = DM(f_de+H#00 );
     MAX_OP;
     DM(f_u+H#00 ) = AR;

{ Defuzzification }
     I0 = ^f_u;
     L0 = %f_u;
     I4 = ^center_u;
     L4 = %center_u;
     call DEFUZ_WC;
     DM( u_HSW ) = AR;
     scaling_output( u_HSW,u_LSW, scale_u);

{ End of fuzzy control-sourcecode }
```

(c) 1993-97 FDG-Systems N. Exler

Use the output value stored in the u_HSW variable to write the crisp value to an output device or use it as an input to a further classical or non-classical module. Include a plant simulation here if you want to simulate the fuzzy module in advance before you try it at the real plant-

```
.ENDMOD;
```

It is also possible to copy all parts of the FPID.DSP file into your application at the appropriate position.

The fuzzy source code above uses subroutines and macros in the essential parts of the fuzzy logic algorithm. The variable choice of the I- and L-registers for different purposes is fixed in the subroutine calls. This is in general no disadvantage. In comparision to a macro implemen-tation it is not possible to change the assignment of the I- and L-registers to the input values, membership values and so on.

General remarks:

There are some necessary definitions for a working algorithm at the beginning of the initial stage.

```
M0 = 0;
M1 = 1;
M4 = 0;
M5 = 1;
ENA AR_SAT;
```

The other M-registers can be used for different increment or decrement values 0 or 1.

After a boot or reboot some program and data memory fields must be initialized before starting the different parts of the fuzzy algorithm.

(c) 1993-97 FDG-Systems N. Exler

- Initialize the *pointer_member* field in the program memory with the number of the membership functions per linguistic variable and the first address of all membership definition fields.
- Initialize the *pointer_f* field in the program memory with the first address of all membership value fields.
- Initialize all *membership value* fields in the data memory with a starting value (normally 0).
- Initialize the *rulebase* with the actual addresses of the fuzzy output variables.

Be careful by using I- and L-registers for other routines than the fuzzy routines. In the text file "fuzzylib.txt" in \FID\FUZLIB the used I-,L-registers for each subroutine are explained. The possible I- and L-registers for

MACRO calls are explained in the macro-librar y "fuztools.lib" in the same directory mentioned above.

(c) 1993-97 FDG-Systems
N. Exler

6. Examples running on EZ-LAB board

6.1 Design of linear and non-linear static fuzzy-elements

This is a guide to designing fuzzy elements with one input and one output (SISO-fuzzy systems) for linear and non-linear characteristic curves. The theoretical background for this design is mentioned in the general description of fuzzy logic in this manual.
Let us start to design a static linear characteristic with a lower and upper limit value. The following declaration for the data format on the ADSP for input values and output values is valid for the whole example. All values for the input and output are normalized to the range of -1 to 1. The format for input and output values is 5.11, and 4.12 for grades of membership.

(c) 1993-97 FDG-Systems N. Exler

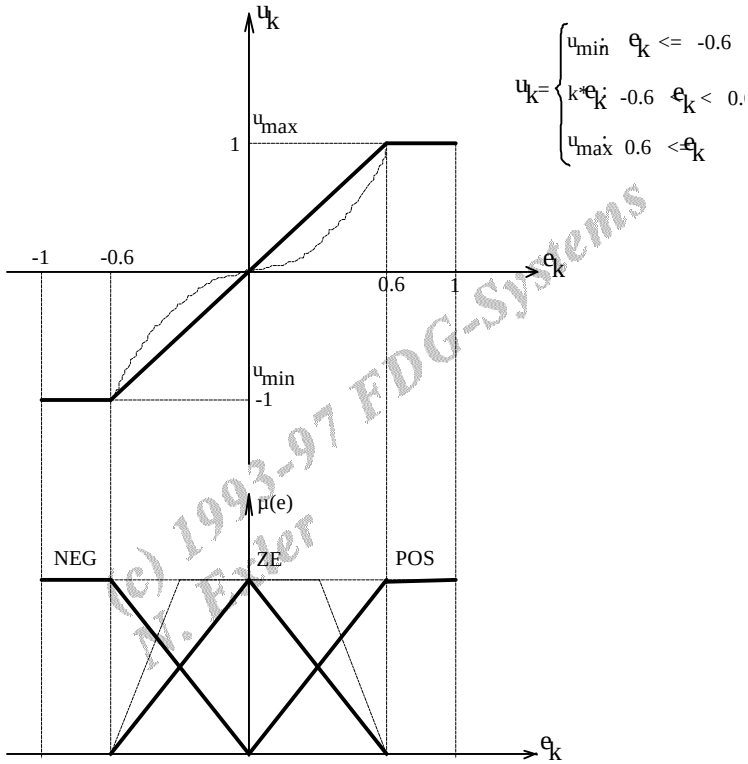


Fig. 17: Membership functions (NEG, ZE, POS) for error e. Source file: STAT_FS.FLC

(c) 1993-97 FDG-Systems N. Exler

The membership functions above can be described as followed:

e_NEG: p_1 = F800, m_1 = 0FFF, p_2 = F800, s_1 = 4000,
p_3 = FB33, m_2 = 0FFF, p_4 = 0000, s_2 = 001A
e_ZE: p_1 = FB33, m_1 = 0000, p_2 = 0000, s_1 = 001A,
p_3 = 0000, m_2 = 0FFF, p_4 = 04CD, s_2 = 001A
e_POS: p_1 = 0000, m_1 = 0000, p_2 = 04CD, s_1 = 001A,
p_3 = 07FF, m_2 = 0FFF, p_4 = 07FF, s_2 = 4000

and the dash-dotted membership function for zero:

e_ZE: p_1 = FB33, m_1 = 0000, p_2 = FE66, s_1 = 0028,
p_3 = 019A, m_2 = 0FFF, p_4 = 04CD, s_2 = 0028

The numerous numbers of degrees of freedom in designing membership functions should be restricted to PI-shaped membership functions (fig. 17). The 3 fuzzy sets above, in combination with the following rulebase and the output fuzzy sets (fig. 19), describe the linear characteristic in fig. 17.
The rules only map the fuzzy sets from the input to the output. For a negative slope the map of the fuzzy sets is just inverted. The rulebase is therefore very simple. In fig. 18 the rules are shown graphically in a kind of Karnaugh-Veitch type diagram.

u \ e	NEG	ZE	POS
	neg	ze	pos

Fig. 18: Rulebase for a positive slope of the linear characteristic.

(c) 1993-97 FDG-Systems N. Exler

for k > 0: if (e_k = **NEG**ative) then (x_k = **neg**ative)
if (e_k = **ZE**ro) then (x_k = **ze**ro)
if (e_k = **POS**itive) then (x_k = **pos**itive)

for k < 0: if (e_k = **NEG**ative) then (x_k = **pos**itive)
if (e_k = **ZE**ro) then (x_k = **ze**ro)
if (e_k = **POS**itive) then (x_k = **neg**ative)

For the loop-mode it is necessary to store the Karnaugh-Veitch type diagram for the fuzzy rules row by row as followed:

k > 0: 0000, 0001, 0002
k < 0: 0002, 0001, 0000

With this set of rules and the following rectangular membership functions for the output, the exact crisp output can be evaluated.

(c) 1993-97 FDG-Systems N. Exler

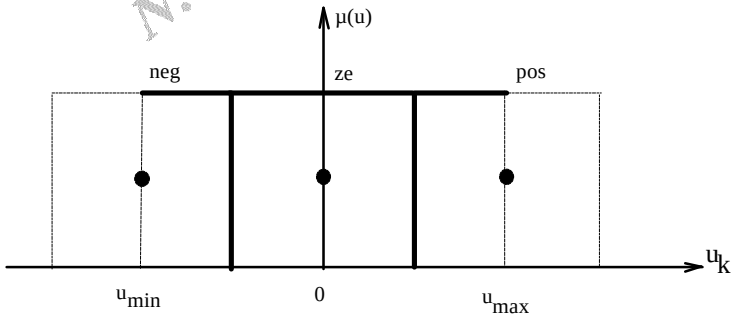


Fig. 19: Output membership functions (neg,ze, pos) with the centroids.

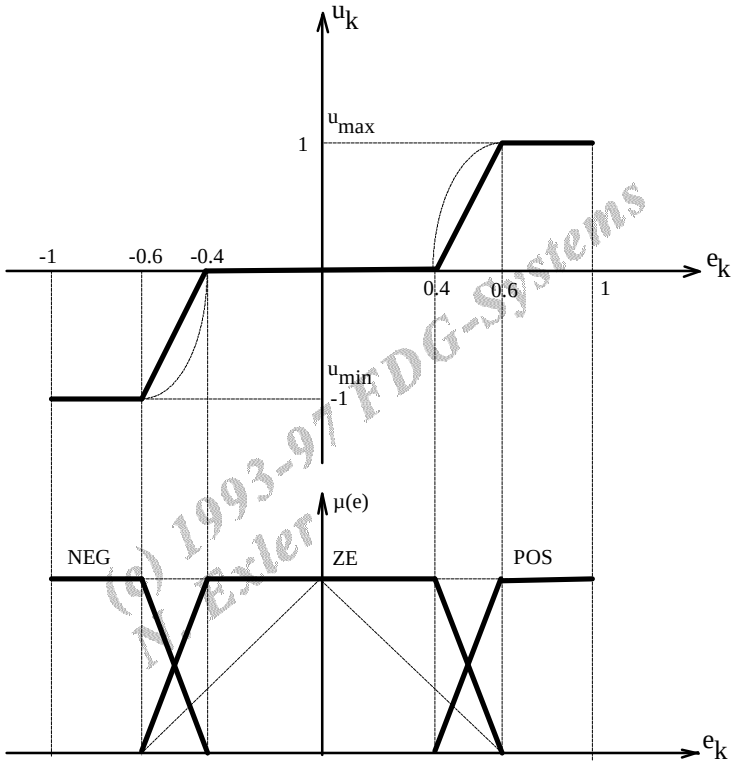
For the simulation of the fuzzy-element on the FID simulation surface it is necessary to define the output fuzzy sets so that the two outermost fuzzy sets are

symmetrical to the maximum (u_{max}) and the minimum output value (u_{min}) respectively. If you want to convert the fuzzy element into ADSP source code, you must restrict the outermost fuzzy sets to their minimum and maximum values (dashed lines in fig. 19).
The necessary centroids for the output Mfs are:

u_neg: F800, u_ze: 0000, u_pos: 07FF

If you want to change the linear characteristic to a non-linear characteristic with dead band, than it is only necessary to change the membership functions for the input values e_k (fig. 20) .

(c) 1993-97 FDG-Systems N. Exler



(c) 1993-97 FDG-Systems N. Exler

Fig. 20: A different set of membership functions (NEG,ZE,POS) for a different characteristic. (Source file: STAT_TNC.FLC)

The membership functions above can be described as followed:

e_NEG: p_1 = F800, m_1 = 0FFF, p_2 = F800, s_1 = 4000,
p_3 = FB33, m_2 = 0FFF, p_4 = FCCD, s_2 = 0050
e_ZE: p_1 = FB33, m_1 = 0000, p_2 = FCCD, s_1 = 0050,
p_3 = 0333, m_2 = 0FFF, p_4 = 04CD, s_2 = 0050
e_POS: p_1 = 0333, m_1 = 0000, p_2 = 04CD, s_1 = 0050,
p_3 = 07FF, m_2 = 0FFF, p_4 = 07FF, s_2 =

4000
and the dash-dotted membership function for zero:

e_ZE: p_1 = FB33, m_1 = 0000, p_2 = 0000, s_1 = 001A,
p_3 = 0000, m_2 = 0FFF, p_4 = 04CD, s_2 = 001A

The simulated characteristic in a 3-dimensional graph looks like this:

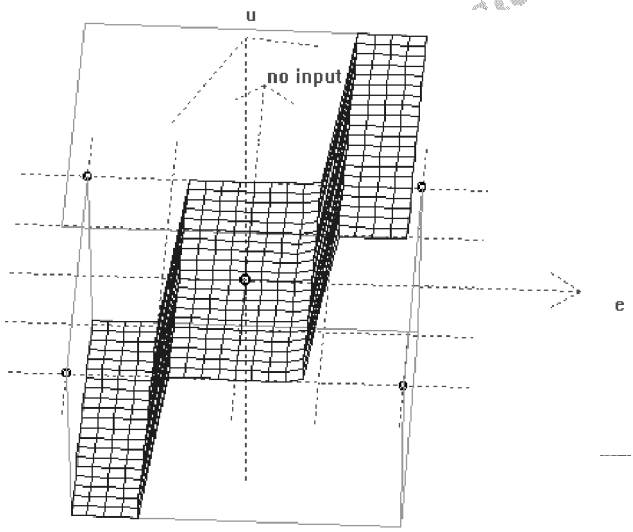


Fig. 21: The non-linear characteristic for the example in fig. 20

The following 4 examples running on EZ-LAB board will demonstrate different functions and characteristics of fuzzy elements in a very simple manner. The outputs of the DAC are used to represent different signals on an oscilloscope. The input signal to the fuzzy element e_k for all examples is a calculated sinus function which can be taken from DAC output 0. Every 50 µs a new

output value will be calculated.

Filename: **stat_fs.exe**

This example displays the three input fuzzy sets on an oscilloscope in x-y mode.
Fuzzy set NEG at output DAC1, ZE at output DAC2 and finally POS at output DAC3.

Filename: **stat_inf.exe**

This example displays the results of the inference mechanism on an oscilloscope on channel 1; channel 0 is always the input signal to the fuzzy element.
Membership value f_u_neg at output DAC1, f_u_ze at output DAC2 and finally f_u_pos at output DAC3.

(c) 1993-97 FDG-Systems N. Exler

Filename: **stat_ki.exe**

This example displays the linear or non-linear characteristic of a fuzzy element. By pressing the IRQ2 button the sign of the slope of the characteristic can be changed. This can be done by just changing the rulebase. If you hold the Flag-In button down when you press the IRQ2 button, then the scale out value is divided by two.

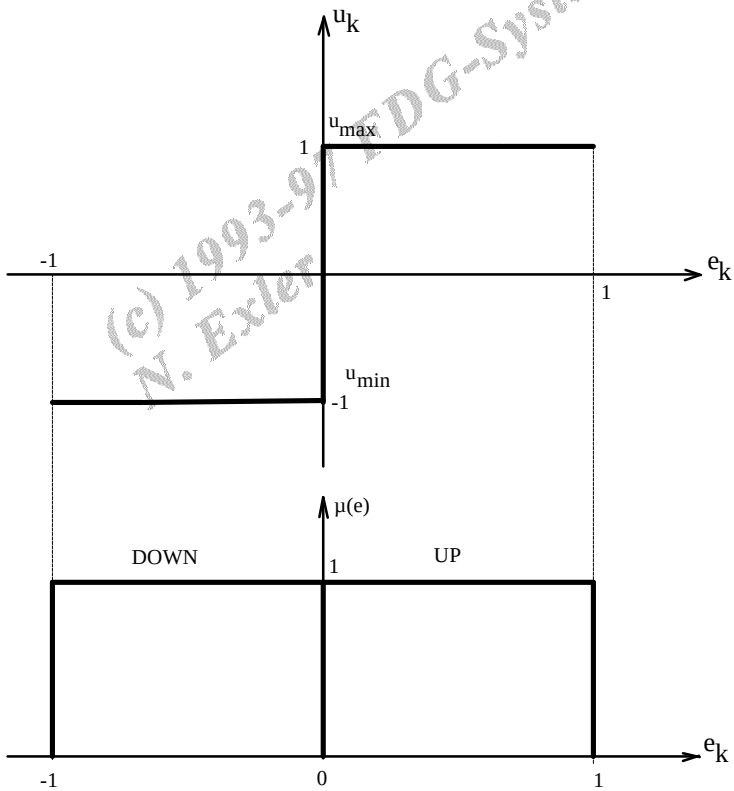
Filename: **stat_tnc.exe**

This example displays two different non-linear characteristics of a fuzzy element. You can toggle between the two characteristics (fig. 17, fig. 20) by pressing the IRQ2 button. In this example it is only necessary to change the shape of the fuzzy sets.

(c) 1993-97 FDG-Systems
N. Exler

6.2 Design of a two-level fuzzy controller

This example demonstrates the design of a two-level fuzzy controller. Designing multiple-level fuzzy controllers afterwards is a quite simple task.
Let us start with a two-step controller without a hysteresis.



(c) 1993-97 FDG-Systems N. Exler

Fig. 22: Membership functions (DOWN, UP) for error e. Source File:TWOP_WH.FLC

The membership functions above can be described as followed:

e_DOWN: p_1 = F800, m_1 = 0FFF, p_2 = F800, s_1 = 4000,
p_3 = 0000, m_2 = 0FFF, p_4 = 0000, s_2 = 4000
e_UP: p_1 = 0000, m_1 = 0FFF, p_2 = 0000, s_1 = 4000,
p_3 = 07FF, m_2 = 0FFF, p_4 = 07FF, s_2 = 4000

From experience it is well known that a two-level controller works like as in the following description. If the error e_k is positive then the output is u_{max}, and when the error e_k is negative then the input is u_{min}. This knowledge enables us to define the rulebase for a fuzzy two-level controller.

IF (e is UP) THEN (u is pos)
IF (e is DOWN) THEN (u is neg)

(c) 1993-97 FDG-Systems N. Exler

For the serialized-mode it is not necessary to define a rulebase like in the examples above.
With this set of rules and the following rectangular membership functions for the output, the exact crisp output can be evaluated.

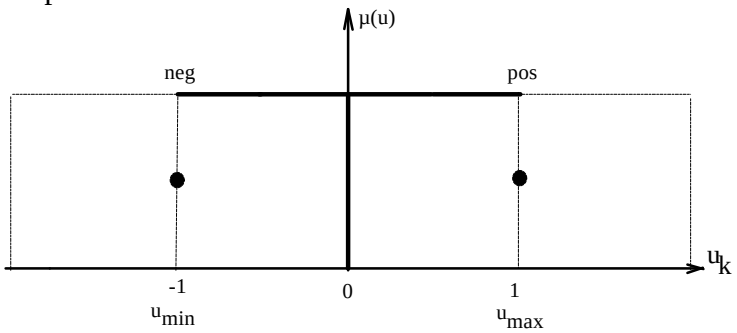


Fig. 23: Output membership functions (neg, pos) with the centroids.

For the output fuzzy sets the same declarations, as in the example before, are valid.
The centroids for the output MFs are:

u_neg: F800, u_pos: 07FF

The two-level controller above has one great disadvantage. It always fulfills a control action by changing its output value. This disadvantage can be reduced by introducing a hysteresis to the two-level controller. The necessary fuzzy sets for the input value e are shown below.

(c) 1993-97 FDG Systems
N. Exler

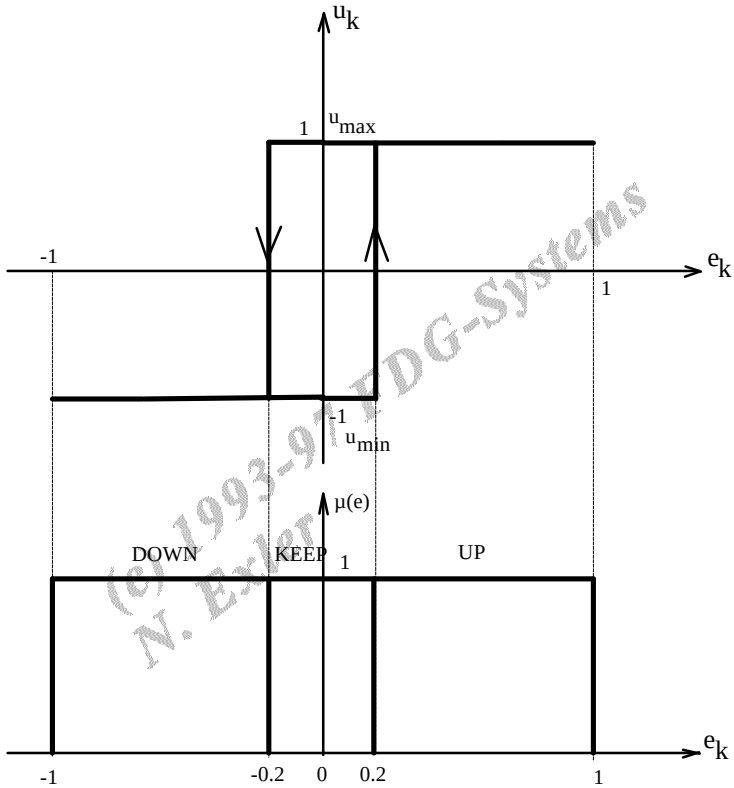


Fig. 24: Membership functions (DOWN, KEEP, UP) for error e. Source File: TWOP_H.FLC

The fuzzy set KEEP works like a holding element which keeps the last output state until another fuzzy set is not equal to zero. The same rulebase can be used as before but we need an IF statement to check if the fuzzy set KEEP is not zero.

Then the rulebase looks like this:

```
    IF      (e is KEEP)   THEN      jump old_state
    IF      (e is UP)     THEN      (u is pos)
    IF      (e is DOWN)     THEN      (u is neg)
old_state: .....
```

This simple change in the rulebase can not be done with the FID Application Software. It is necessary to do this change manually in the ADSP source code. The conclusion of all this is that you can have very simple rulebases by just introducing non-classical fuzzy IF-THEN rules.

(c) 1993-97 FDG-Systems N. Exler

The two-level fuzzy controller with a hysteresis can be used to control a plant with PT1 behaviour. The control loop has the following structure:

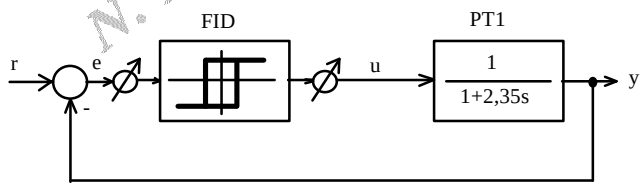


Fig. 25: Closed-loop control structure with a fuzzy two-level controller.

The following 3 examples running on an EZ-LAB board demonstrate different functions and characteristics of two-level fuzzy controllers. The outputs of the DAC are used to represent different signals on an oscilloscope. The input signal to the fuzzy element e_k for the first and second example is a calculated sinus function which can be taken from

DAC output 0. Every 50 µs, a new output value will be calculated.

Filename: **twop_wh.exe**

This example displays the characteristic of a two-level controller without a hysteresis on an oscilloscope. The best display mode is the x-y mode (DAC0:x,DAC1:y).

Filename: **twop_h.exe**

This example displays the characteristic of a two-level controller with a hysteresis on an oscilloscope. The best display mode is the x-y mode(DAC0:x,DAC1:y).

Filename: **twop_pt1.exe**

This example displays the step response of the closed loop control structure in fig. 25. With the IRQ2 button you can toggle between two reference values, and by pressing the Flag-In button and IRQ2 button together, the width of the KEEP fuzzy set is changed..

(c) 1993-97 FDG-Systems N. Exler

6.3 Design of a simple fuzzy PI-controller compared to a classical PI-controller

For this example a plant with two poles in the left half s-plane has been chosen. The plant has the following values for the time constants T_1 = 1s and T_2 = 0,5s and a gain of G_p = 1,35. This plant should be controlled by a very simple fuzzy PI-controller and then compared to an optimally designed classical PI-controller.

The control loop has the following structure:

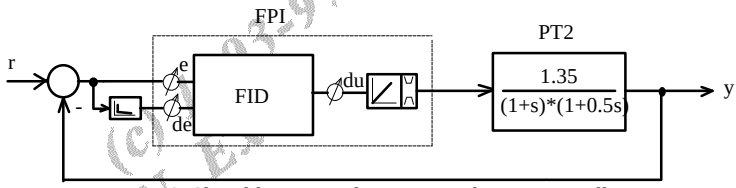


Fig. 26: Closed-loop control structure with a FPI-controller.

The input values of the FPI-controller are the error between the reference r and the controlled value y and the change of error. The pure fuzzy module (FID-module) then has a PD-behaviour. If an integrator is used after the defuzzified output, with constraints on the upper and lower value, it will lead to a PI-behaviour. The same dynamic behaviour could be attained by using the error and the integral of the error as input values.

Both input values could be changed by using scaling factors (scaling values from -16 to 16 are possible). For

(c) 1993-97 FDG Systems N. Exler

this purpose a scaling macro or a scaling subroutine can be used.

☞ For simulation make sure that the following values and connections are made:

Sampling time: 50ms

Connection list:

e	1*step - 1* out
de	4* dif(e)

Source Files: FIRST.FLC, PT2_EX3.PLN

(c) 1993-97 FDG-Systems N. Exler

Two membership functions are selected for each input value and are shown in fig. 27.

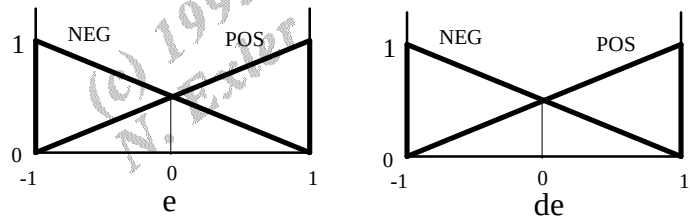


Fig. 27: MFs for error e and change of error de.

The necessary membership functions are defined as followed:

e_NEG, de_NEG:

p_1 = F800, m_1 = 0FFF, p_2 = F800, s_1 = 4000,
p_3 = F800, m_2 = 0FFF, p_4 = 07FF, s_2 = 0008

e_POS, de_POS:

p_1 = F800, m_1 = 0000, p_2 = 07FF, s_1 = 0008,
p_3 = 07FF, m_2 = 0FFF, p_4 = 07FF, s_2 = 4000

The number of membership functions for the fuzzy output of the FID-controller chosen is 3. For these MFs of the fuzzy output, just the centroids of the MFs are stored (see fig. 28).

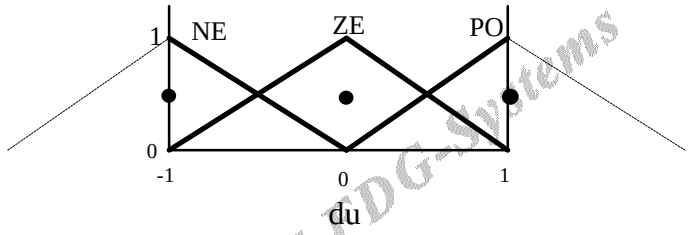


Fig. 28: Output MFs with the centroids.

(c) 1993-97 FDG-Systems N. Exler

If the output MFs do not have the same size then this can be taken into consideration by storing the scaled areas and the centers of the MFs.

With two MFs (fuzzy sets) for each input variable, the maximum number of the linguistic rules is 2 x 2. These rules can be graphically shown in a kind of Karnaugh-Veitch type diagram, which has the following structure:

		e	
	du	NE	PO
de	NE	NE 0	ZE 1
	PO	ZE 1	PO 2

Fig. 29: Karnaugh-Veitch type diagram for the fuzzy rules.

If the rules are written in a linguistic form then the 4 rules are as follows:

IF (e is Negative) AND (de is NEgative) THEN (du is NEgative)
IF (e is Negative) AND (de is POsitive) THEN (du is ZEro)
IF (e is POsitive) AND (de is NEgative) THEN (du is ZEro)
IF (e is POsitive) AND (de is POsitive) THEN (du is POsitive)

(c) 1993-97 FDG-Systems N. Exler

In order to implement the rulebase numerically it is necessary to number the output membership functions from zero to the highest number. i.e. NE=0, ZE=1, PO=2.
For the loop-mode it is necessary to store the Karnaugh-Veitch type diagram for the fuzzy rules row by row.
i.e. rulebase: 0, 1, 1, 2
The rulebase needs 4 program memory locations and these values will be incremented during the initial stage by the start address of the field for the fuzzy output values. Therefore the whole code and data for fuzzy controllers is fully relocatable in program and data memory.
Mamdani's Max-Min inference method is used for the evaluation of the rulebase and the resulting output membership values which are stored in a field with 3 data memory locations.

By choosing a defuzzification method for the evaluation of a crisp output value the change in a manipulated variable can be derived. The simplest defuzzification method is the first type with singletons representing the centroids of the membership functions for the fuzzy output. An accurate defuzzification uses rectangular membership functions for the evaluation of the crisp output value.

The simulated control map for this example looks like this:

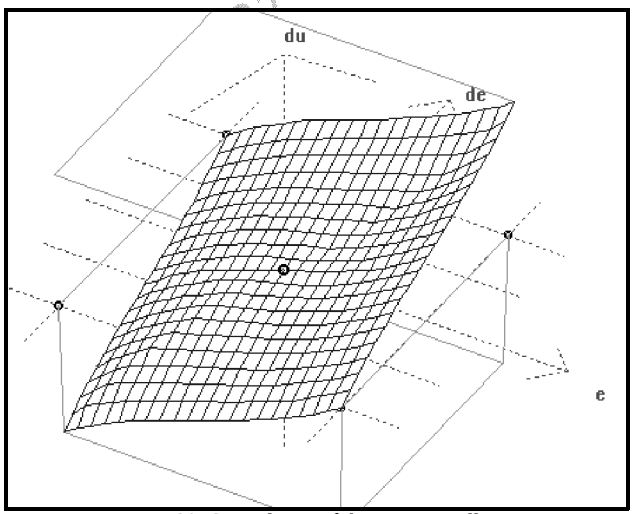


Fig. 30: Control map of the FPI-controller

In this case, when we know exactly the behaviour of the plant in form of a mathematical description, it is possible to compare the fuzzy PI-controller to a

classical PI-controller. An optimal classical PI-controller can be designed by a criterion with the restriction of 5% overshoot. The result for such an optimization for the plant above is:

$T_N = T_1 = 1\ s$
$K_r = T_N / (2*T_2*G_s) = 0{,}74074$

In continuous form the formula can be written as:

$C(s) = K_r*(1 + sT_N) * 1 / sT_N = U(s) / E(s) =$
$dU(s) / E(s) * U(s) / dU(s)$

in discrete form by using the seperation before:

$u(kT) = (e(kT) + T_N*(e(kT) - e((k-1)T)) / T_s)* T_s*K_r / T_N * \sum du(kT)$

This formula can be used to demonstrate the connection between a FPI-controller with a control map quite similar to the control map of a classical PI-controller. The values calculated out of this formula for the scaling values can be used as a good starting point for further optimization of the FPI-controller.

$se = 1, \quad sde = T_N / T_s = 20,$
$sdu = T_s*K_r / T_N = 0{,}03704$

This FPI-controller has a very similar behaviour to a classical PI-controller. The reasons for this are the very simple rulebase and two triangular membership functions . The simulation results for the classical and the fuzzy PI-controller are shown in fig. 31.

(c) 1993-97 FDG-Systems N. Exler

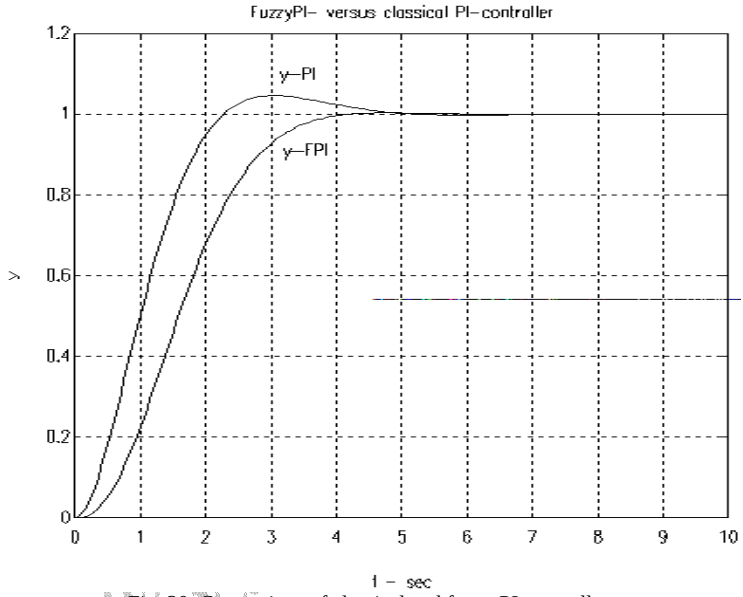


Fig. 30: Comparison of classical and fuzzy PI-controller.

The following 3 examples running on an EZ-LAB board will demonstrate different functions and characteristics of two-level fuzzy controllers. The outputs of the DAC are used to represent different signals on an oscilloscope. The input signal to the fuzzy element e_k for the first and second example is a calculated sinus function which can be taken from DAC output 0. Every 50 µs a new output value will be calculated.

Filename: **first_r.exe**

This example displays the control map of the FPI-controller on an oscilloscope (x-y mode). Input e

(DAC0) of the FPI-controller (output on DAC1) is fed by a sinus-function, whereas input de is kept constant. With IRQ 2 it is possible to change this value in steps from its minimum to its maximum. By pressing Flag-In and IRQ2 button together it is possible to swap the step value between the minimum and the maximum.

Filename: **first_s.exe**

This example displays the step response of the closed loop control structure in fig. 26. With the IRQ2 button you can toggle between two different inference methods. (MAX-MIN, Alg-Prod - MIN)

Filename: **first_h.exe**

This example displays the step response of the FPI-controller defined before.

(c) 1993-97 FDG-Systems N. Exler

6.4 Design of a simple fuzzy PID-controller

The most commonly used controller is a PID-controller. This example describes the design of a very simple fuzzy PID-controller with only 3 rules. The input values of the FPID-controller are the error between the reference r and the controlled value y, the sum of the error and the change of error. The pure fuzzy module (FID-module) then has a PID-behaviour. The same dynamic behaviour could be attained by using the error, the change of error and the 2nd change of error as input values and an integration element used after the defuzzified output.

(c) 1993-97 FDG-Systems N. Exler

All input and output values can be changed by using scaling factors (scaling values from -16 to 16 are possible). For this purpose a scaling macro or a scaling subroutine can be used.
The following structure (fig. 32) represents one possibility for a FPID-controller.

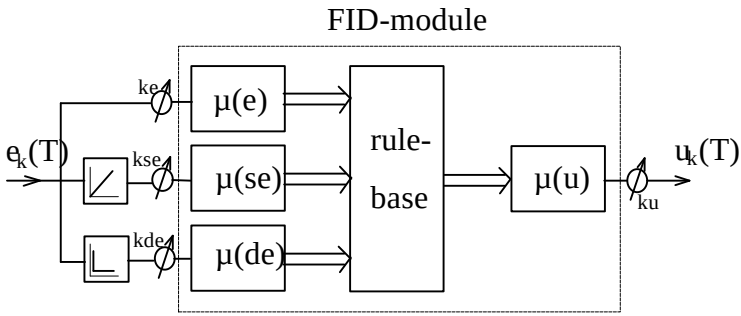
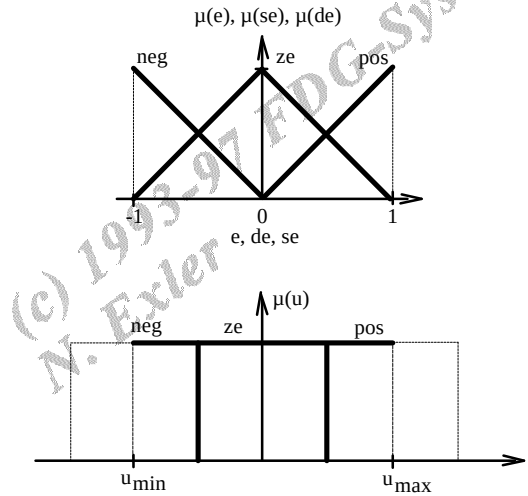


Fig. 31: Fuzzy controller with PID behaviour.

The inputs and the output in fig. 32 are defined as followed:

$e = e_k$, $se = e_i$, $de = e_k - e_{k-1}$, $u = u_k$

For demonstration purposes all three inputs have the same shape and universe of discourse. With the different scaling values for each input it is possible to change the universe of discourse.



(c) 1993-97 FDG-Systems N. Exler

Fig. 32: Input and output fuzzy sets for the FPID-controller. Source File: FPID.FLC

With three MFs (fuzzy sets) for each input variable, the maximum number of the linguistic rules is 3 x 3 x 3. These rules can be graphically shown in a kind of 3-dimensional Karnaugh-Veitch type diagram, which has the following structure:

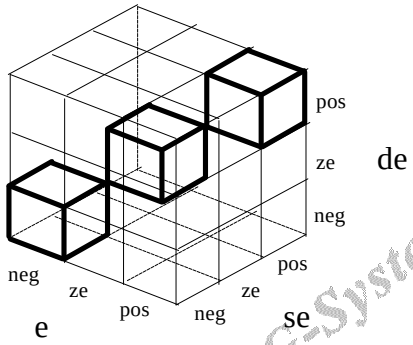


Fig. 33: Karnaugh-Veitch type diagram for all fuzzy rules.

(c) 1993-97 FDG-Systems N. Exler

It is not necessary to use all rules for a working FPID-controller. A very good result can be reached by using only three rules out of 27 rules ! These three rules are marked in bold boxes in the 3 dimensional KV-type diagram above. Using all rules means that it is possible to tune the FPID controller for special behaviour like good disturbance suppression.
The rulebase used can be written as:

IF (e is pos) OR (se is pos) OR (de is pos) THEN (u = neg)
IF (e is ze) OR (se is ze) OR (de is ze) THEN (u = ze)
IF (e is neg) OR (se is neg) OR (de is neg) THEN (u = pos)

Normally the AND-operator is used for the antecedents, but here the OR-operator is a very good alternative and quite effective.

The formula for a PID-controller in the continuous case is:

$$R(s) = K_R + 1/(sT_N) + sT_V = U(s) / E(s)$$

in discrete from:

$$1*u(k) = K_R*e(k) + T_s/T_N * \sum e(k) + T_V/T_s * (e(k) - e(k-1))$$

This formula can be used to demonstrate the connection between a FPID-controller with a control map quite similar to the control map of a classical PID-controller. The values calculated out of this formula for the scaling values can be used as a good starting point for further optimization of the FPID-controller.

$$ke = T_R, \quad kde = T_V / T_s, \quad kse = T_s / T_N, \quad ku = 1.$$

It is possible to use this formula to interpret the universe of discourse of the input fuzzy sets in a different way than in fig. 33 above (fig. 35).

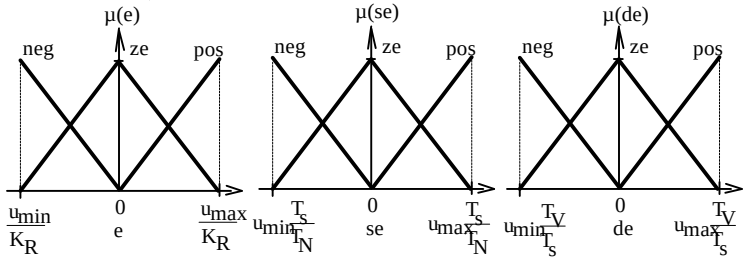


Fig. 34: Input fuzzy sets for the FPID-controller.

The step response of a FPID-controller is quite similar to a step response of a classical PID-controller (fig. 36). By changing the fuzzy sets for the inputs and the output, the behaviour can be changed quite a lot. We

(c) 1993-97 FDG-Systems N. Exler

can say that the FPID-controller is a kind of non-linear controller whereas the linear PID-controller is just a special case.

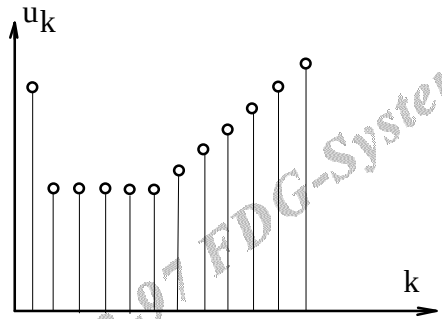


Fig. 35: Step response of a FPID-controller.

(c) 1993-97 FDG-Systems N. Exler

The following example on the EZ-LAB board demonstrates the behaviour of the FPID-controller explained above.

Filename: **fpid_h.exe**

In this example the step response of the FPID-controller is displayed on an oscilloscope.

DAC0: Fuzzy PID Output
DAC1: error signal
DAC2: sum of error signal
DAC3: difference of error signal

7. Introduction to Fuzzy Logic

7.1 Overview

Fuzzy logic is a generalization of binary logic (Boolean logic), and therefore, a fuzzy logic based system can completely represent a crisp logic system. The converse is not true. Fuzzy logic is based on fuzzy set theory and provides a rigorous framework for representing non-crisp situations. Similarly, the fuzzy set theory is a generalization of classical set theory.
In 1965, Lofti A. Zadeh, a professor of electrical engineering at the University of California, Berkeley expanded the concept of a classically defined set to that of a fuzzy set.
The basic fuzzy rule-based structure can be used in many different types of applications, including control, process control, decision making, scheduling, prediction, and estimation. By allowing high flexibility in the definition of fuzzy logic operations, and especially in how the cobination of the firing strength of all rules and the defuzzification is performed, the area of applications is even further increased.

(c) 1993-97 FDG-Systems N. Exler

7.2 Fundamentals of Fuzzy Logic

Unlike binary logic and binary algebra used in all standard computers, fuzzy logic is multivalued. Instead of an element being a member of a set or not, fuzzy logic allows degrees of membership so that something

can be partially true and partially false at the same time or in other words, something can be a member of two or more fuzzy sets. Fuzzy sets which are therefore an essential part of the fuzzy set theory, can be defined as uncertain quantities of objects or values. First of all the meaning of uncertain quantities (fuzzy sets) will be explained, starting with crisp quantities.

(c) 1993-97 FDG-Systems N. Exler

7.2.1 Fuzzy sets and membership functions

The separation of people into groups (sets) according to their height can give an idea of crisp and uncertain (fuzzy) sets. If, for example, three crisp sets { *small, medium, tall*} are defined and the appropriate people are allocated to them, classical set theory (crisp sets) demands the definition of two thresholds for three sets. These sets are assigned by the system designer, and are given labels such as *small, medium, tall.*

e.g.: threshold 1: 1,70 m;
threshold 2: 1,80 m

The three sets have then the following ranges:

small < 1.70m
1.70m ≤ medium < 1.80m
1.80m ≤ tall

This is shown graphically in figure 36. The circles around the ranges along the height axis are just to visualize these ranges as in the classical set theory. Six people with different heights are marked along the height axis (person A: 1.65m; person B: 1.85m, person C: 1.73m, person D: 1.78m, person E: 1.70m, person F:

1.80m).

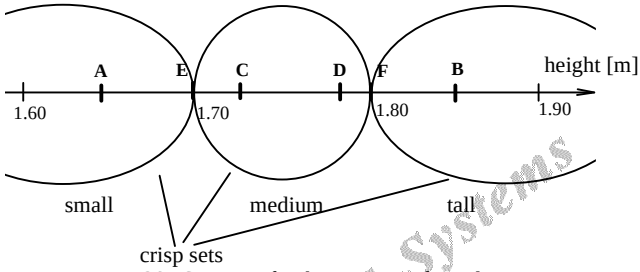


Fig. 36: Crisp sets for three groups of people.

(c) 1993-97 FDG-Systems N. Exler

The degree (TRUE or FALSE) of membership μ to a crisp or fuzzy set {*small, medium, tall*} can be described by using membership functions (see figure 37). For crisp sets it isn't really necessary to define membership functions, but it makes it easier to understand the difference between membership functions and crisp or fuzzy sets later on.

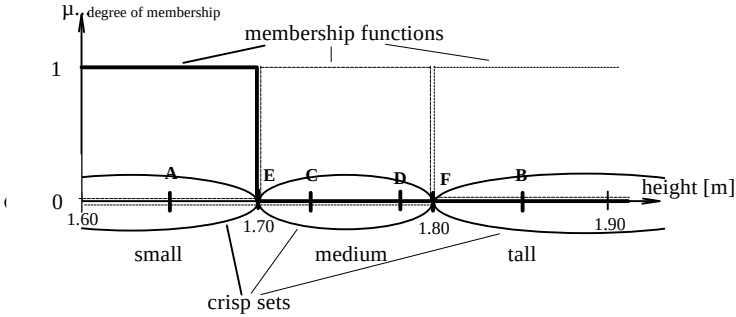


Fig. 37: Membership functions describing the degree of membership μ of crisp sets.

The great disadvantage of this division (see figure 36 and 37) is that a person with a height of 1,79 m belongs

to the group of medium people, but a person who is just 1 centimeter taller than the person before belongs to the group of tall people. Normally we would say that the second person is just a little taller than the first person. One solution could be the definition of more sets, but the discrimination between the sets remains still very crisp. The division of human beings and also of nature in general is much smoother than in the Boolean logic and therefore multi-levels of separation are used (e.g. in the brain four different signal levels are known).
The only solution to this problem is uncertain quantities called fuzzy sets in the fuzzy set theory. The fuzzy set theory can be seen as a generalization of the ordinary set theory. To extend the crisp sets to fuzzy sets it is necessary to overlap the crisp sets (see figure 38). The overlapping region is the fuzzy part of the overlapping sets.

(c) 1993-97 FDG-System N. Exler

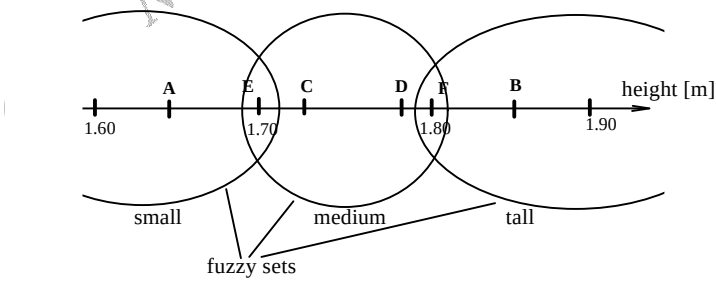


Fig. 38: Fuzzy sets for three groups of people.

This overlapping region of the fuzzy sets can be described by using membership functions (MFs) with different shaped edges. e.g.: In figure 39 trapezium shaped MFs (straight lines with a finit slope) are used to

describe fuzzy sets. The main other shapes used are triangular, Gauß- and cos-shaped MFs.

With this concept a person with a height of 1,79 m belongs with a degree of 0.75 to the fuzzy set *medium* and with a degree of 0.25 to the fuzzy set *tall*. In a linguistic interpretation this means, this person is a little bit more medium height than tall. It is now clear that the fuzzy concept describes the decision making process of humans much better than the classical mathematical concepts.

(c) 1993-97 FDG-Systems N. Exler

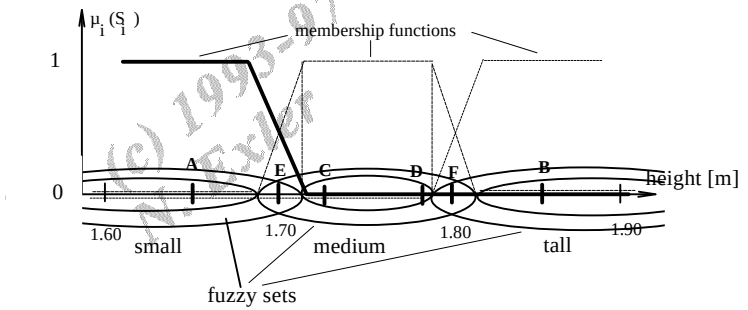


Fig. 39: Membership functions for fuzzy sets.

Now, it is necessary to give some definitions before going further on. The fuzzy sets are described by functions called membership functions as already defined in figure 37 and 39 which describe the degree of membership of every input value to a fuzzy set between the values 0 and 1 (or 0 and 100 %). In this case the fuzzy sets are normalized to the range of 0 to 1.
Mathematically the fuzzy sets can be described in different kinds of definitions. The most common

definitions are:

A fuzzy set ***A*** in ***X*** is a set of ordered pairs:

$$\boldsymbol{A} = \{ (x,\mu_{\boldsymbol{A}}(x)) \mid x \in \boldsymbol{X} \}$$

where $\mu_{\boldsymbol{A}}(x)$ is the membership function of x in ***A***, which maps the collection of objects in the range of definitio n ***X*** (universe of discourse) to the membership space ***M***.

That means in discrete form for the fuzzy set mediu m before:

medium = { (1.68,0.0), (1.70,0.5), (1.72,1.0), (1.74,1.0), (1.76,1.0), (1.78,1.0), (1.80,0.5), (1.82,0.0) | height ∈ (1.50 .. 2.50m) }

and in continous form:

$$\text{medium} = \begin{cases} 0.0 & \text{height} \le 1.68\ \text{m} \\ 25*\text{height}-14.88 & 1.68\ \text{m} < \text{height} < 1.72\ \text{m} \\ 1.0 & 1.72\ \text{m} \le \text{height} \le 1.78\ \text{m} \\ -25*\text{height}+14.88 & 1.78\ \text{m} < \text{height} < 1.82\ \text{m} \\ 0.0 & \text{height} \le 1.82\ \text{m} \end{cases}$$

(c) 1993-97 FDG-Systems N. Exler

Beside the first definition the following definition i s quite common:

$$\boldsymbol{A} = \mu_{\boldsymbol{A}}(x_1) / x_1 + \mu_{\boldsymbol{A}}(x_2) / x_2 + \ldots\ldots = \sum_i \mu_{\boldsymbol{A}}(x_i) / xi \quad i = 1 .. n$$

or

$$\boldsymbol{A} = \int_x \mu_{\boldsymbol{A}}(x) / x$$

That means in discrete form for the fuzzy set medium:

$$\boldsymbol{A} = 0.0/1.68 + 0.5/1.70 + 1.0/1.72 + \ldots\ldots$$

Further definitions and examples about fuzzy sets can be

found in numerous books and papers describing fuzz y logic (see chapter 8).

7.2.2 Shape of membership functions

The shape of membership functions has an influence on the "weighting" of the input values in the range o f definition. e.g. A triangular shaped MF has only on e input value where the de gree of membership (DOM) μ is 1. If it is necessary to have a range of input values where the DOM values should be 1 it is necessary to us e trapezium shaped MFs. Gauß- shaped MFs are often used for statistically clustered input values. For contro l purposes just Π-shaped MFs are used to map the inpu t values to the fuzzy sets.
In the example of chapter 7.2.1 just rectangular an d trapezium shaped membership functions (MFs) hav e been taken into consideration. These shapes of MFs are called Π-shaped. Special cases of Π-shaped MFs ar e Λ-shaped (or triangular MFs), Z-shaped and S-shape d MFs (figure 41). Other shapes for membership functions which are often used are Gauß- and cos-shaped MF s (figure 41 and 42). All these MFs can be represente d only by four corner points p_1, p_2, p_3, p_4. and the shape of MF. The values betwee n these points can be interpolated very easily.
For fuzzy outputs a more simpler representation of MFs are often used. The used "Mfs" are singletons of rea l membership functions (figure 43). which can b e described by one or two values. The first value is always a crisp value in the output definition range (universe o f

discourse) and is called c (cen ter) and the second value is a weighting on that singleton (a...area). The weightin g value a (can be understood as the area under the MF) describes the influence of a singleton on the crisp output value compared to the other singletons.

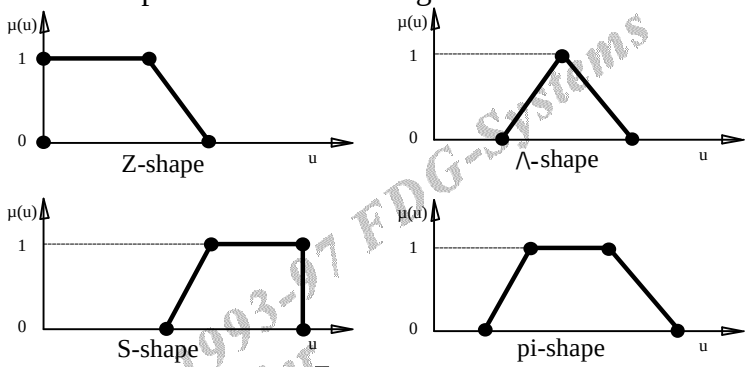


Fig. 40: Different Π-shaped membership functions.

(c) 1993-97 FDG-Systems N. Exler

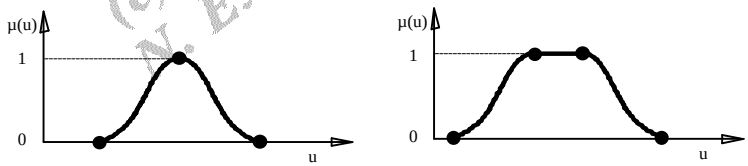


Fig. 41: Gauss-shaped membership functions.

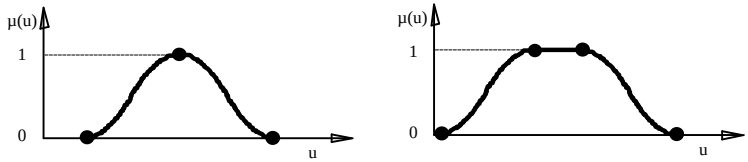


Fig. 42: Cos-shaped membership functions.

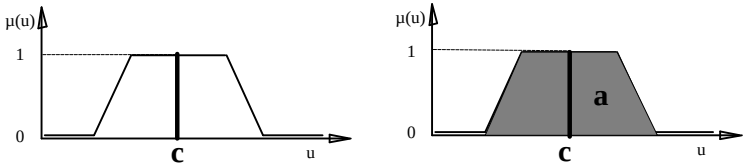


Fig. 43: Singletons for membership functions

In practice, the system designer choose the membership functions and the fuzzy sets are a result of this choice . The same labels which are used for the membershi p functions are used for the fuzzy sets and vice versa. Each of the labels represent a fuzzy set positioned in th e operational domain of possible crisp values.
The representation in form of numbers can be therefor e simplified quite a lot using Π-shaped membershi p functions with only 4 corner points, and a linea r interpolation between these points. Another possibilit y for the representation of the membership functions with arbitrary shapes could be the representation by arrays of numbers, but that consumes a lot of memory spac e Therefore, PI-shaped MFs are well suited for practica l implementations on microprocessors especially DSP's.

(c) 1993-97 FDG-Systems N. Exler

7.2.3 Set operators

Up to now, the differences between fuzzy sets an d membership functions are described. Therefore th e membership function is the mo st important part for fuzzy set operations. Operations with fuzzy sets (uncertai n sets) are defined via their membership functions.
The basic operators for the Boolean logic are AND, OR

and NOT and these operations can also be used for se t operations. But these basic operators are just used fo r fuzzy set operations in order to describe the problems in a linguistical form. For the actual calculation of fuzz y values the basic operators AND and OR are substitute d by different set operators.
The most used set operators for fuzzy contro l applications are the minimum an d the maximum operator and sometimes the complement operator (negation) . These operators were origionally used by Zadeh an d Mamdani.

(c) 1993-97 FDG-Systems N. Exler

Intersection of 2 fuzzy sets: $\boldsymbol{C = A \cap B}$
can be described by the minimum operator:
$\boldsymbol{\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\}}$ $x \in \boldsymbol{X}$
Union of 2 fuzzy sets: $\boldsymbol{C = A \cup B}$
can be described by the maximum operator:
$\boldsymbol{\mu_C(x) = \max\{\mu_A(x), \mu_B(x)\}}$ $x \in \boldsymbol{X}$
Complement of a fuzzy set: $\boldsymbol{A' \Leftarrow A}$
can be described by the complement operator:
$\boldsymbol{\mu_{A'}(x) = 1 - \mu_A(x)}$ $x \in \boldsymbol{X}$

e.g.: $\mu_A(x) = 0.5; \mu_B(x) = 0.8$
$\boldsymbol{A \cap B}$ $\min\{0.5; 0.8\} = 0.5$
$\boldsymbol{A \cup B}$ $\max\{0.5; 0.8\} = 0.8$
$\boldsymbol{A'}$ $\mu_{A'}(x) = 1 - \mu_A(x) = 0.5$
The binary AND-operator and the min-operator are very closely related to each other. Similarly, the binary O R operator and the ma x-operator are very closely related to each other. The reason for the first relation is that th e result of an AND-operation is only 1 if both or all input

values are 1. For the set operator minimum the same conclusion can be used. The same relation is valid for the OR and the maximum operator.
The three basic fuzzy operations are graphically shown in figure 46.

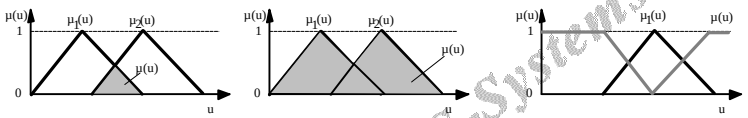


Fig. 46: Minimum, maximum and complement operation

(c) 1993-97 FDG-Systems N. Exler

The following mathematical characteristics are valid for the min- and max-operators:

commutative: $op(\mu_A, \mu_B) = op(\mu_B, \mu_A)$

associative: $op(\mu_A, \mu_B, \mu_C) = op(op(\mu_A, \mu_B), \mu_C)$

This two characteristics are mainly important if more than two input values are used in a linguistical rule.

If the min-operator is compared to human decision making then it can be interpreted as a pessimistic operator which always takes the smallest value. Therefore, this operator has no compensation feature. Sometimes, this feature is desirable, but on the other side it is often an undesirable feature. e.g. People compensate some weaknesses in certain areas by their strengths in other areas. The max-operator is very optimistic and therefore it is an extremely compensative operator.
Besides the min- and max-operator a lot of other operators can be used for the AND and OR operation. (e.g.: algebraic product, algebraic sum, drastic product, drastic sum, ..)

(c) 1993-97 FDG-Systems
N. Exler

7.3 Structure of a fuzzy system

All fuzzy logic systems use a rule-base (knowledge base) as their central structure. Rules, typically cast in an IF ... THEN ... syntax, represent system operation and mapping inputs to outputs. Measured and calculated crisp input values are fuzzified, using membership functions, into fuzzy truth values (or degrees of membership). These are then applied as conditions to the rules contained in the rule-base, with triggered rules specifying necessary actions, again as fuzzy truth values. These actions are combined and defuzzified into crisp, executable system outputs. Where inputs and outputs are continuous (as in control applications), this fuzzification-inference-defuzzification process is performed on an ongoing basis, at regular sampling intervals.

(c) 1993-97 FDG-Systems N. Exler

Conceptually this process is similar to the use of a Fast Fourier Transform (FFT), to transform time domain signals into the frequency domain, to process the resulting frequencies, and then to transform the results back into time domain. The added expense of transforming from the time to the frequency domain is justified because the system model is easier to understand and to manipulate in terms of frequencies.
Similarly, a fuzzy system "transforms" signals from the "crisp domain" to the "fuzzy domain", makes decisions based on these fuzzy values and a knowledge of the desired system operation cast in fuzzy terms (rules), and then transforms the results back into the crisp domain for execution (figure 47). The justification is, as with

frequency domain processing, that the system model is easier to understand and to manipulate in the fuzzy domain than in the crisp domain.

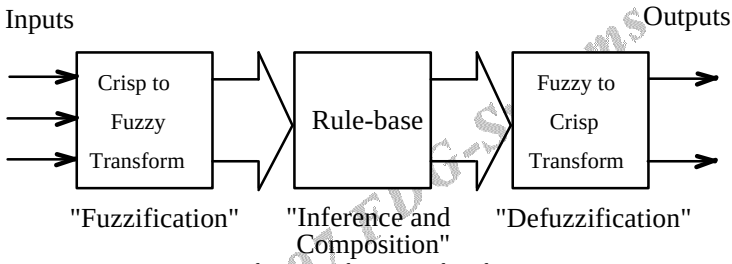


Fig. 47: Fuzzification-Inference-Defuzzification process

This basic fuzzy rule-based structure can be used in many different types of applications, including control, process control, decision making, scheduling, prediction, and estimation. By allowing high flexibility in the definition of fuzzy logic operations, and especially in how the combination of the firing strength of all rules and the defuzzification is performed, the area of applications is even further increased.

7.3.1 Fuzzification

Consider a fuzzy temperature controller with two inputs. The two inputs are the temperature T and the change in temperature ΔT. The actual measured temperature T and the calculated change in temperature $\Delta T = T_{old} - T$ are crisp values.

For a classical temperature controller (e.g. PID) only the error in temperature is used as an input. The output of this PID-controller is the actual control action for a control unit (e.g. heating system).
If a fuzzy temperature controller, represented by fuzzy terms, is used, then it is necessary to convert (or transform) the current inputs into fuzzy inputs by finding the degrees of membership (fuzzy truth values) for all input membership functions.
This conversion from the crisp input space into the fuzzy input space is called fuzzification. The two input variables T and ΔT are called linguistic variables, and each of these variables consist of a few membership functions (fuzzy terms). The membership functions (MFs) are assigned by the system designer (figure 48), and are given labels such as *cold, warm* for T and *negative, zero, positive* for ΔT. Each of these labels represent a fuzzy set positioned in the operational domain (universe of discourse) of possible crisp values.

In figure 48 the fuzzification process for input values of T = 10°C and ΔT = -2°C/time unit (shown on the x-axis). The degree of membership is the grade value at the intersection the system input value makes with a membership function.

(c) 1993-97 FDG-Systems N. Exler

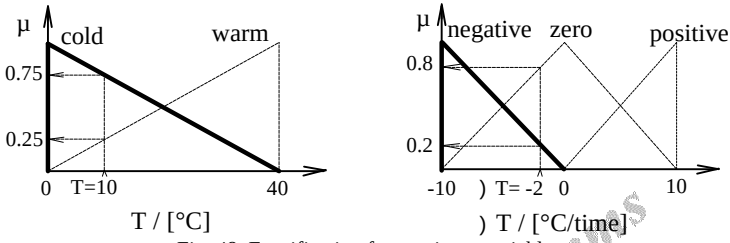


Fig. 48: Fuzzification for two input variables.

In figure 48 this yields a fuzzy input value for T of 0.75 for MF 'cold', and of 0.25 for MF 'warm'. This process is repeated for the change of temperature input yielding 0.2 for MF 'negative', 0.8 for 'zero', and zero for MF 'positive'.

(c) 1993-97 FDG-Systems N. Exler

7.3.2 Inference and Composition

The inference processing (rule evaluation) is the central part of the knowledge based decision making, and is expressed by linguistical rules. Rules are statements expressing a dependancy relation among system inputs and system outputs. Individual rules represent parts of the solution to a problem. All rules considered together determine the final solution.
Rule evaluation takes the fuzzy inputs (degrees of membership) from the fuzzification step and the rules from the knowledge base and calculates fuzzy outputs. A typical rule base for a temperature controller below shows in principle the rule evaluation. The membership functions for the output heating are shown in figure 47.

	RULE					STRENGTH
IF	T is cold (0.75)	AND	ΔT is zero (0.8)	THEN	heating is high	(0.75)
IF	T is cold (0.75)	AND	ΔT is negative (0.2)	THEN	heating is high	(0.2)
IF	T is cold (0.75) (0.0)	AND	ΔT is positive (0.0)	THEN	heating is medium	
IF	T is warm (0.25) (0.2)	AND	ΔT is negative (0.2)	THEN	heating is medium	
IF	T is warm (0.25)	AND	ΔT is zero (0.8)	THEN	heating is off	(0.25)
IF	T is warm (0.25)	AND	ΔT is positive (0.0)	THEN	heating is off	(0.0)

Next to each of the membership labels in the IF-parts (antecedants) the corresponding fuzzy inputs obtained in the fuzzification step are shown in brackets. To the right of each rule the resulting firing strength of the rules are shown. These values can be calculated by using a set operator like the min-operator for the AND operation.
If more than one rule fires at the same fuzzy output, then the rule that is most true will dominate. This can be implemented by taking the maximum of these rule strengths and assigning the value to the corresponding fuzzy output MFs.
This inference method is called MIN-MAX inference. i.e. Take the minimum of each condition (antecedant) and then combine all output values which are assigned to the same output MF by using the max-operator. Normally, these resulting values are applied to the output MFs by using Mamdani's minimum operation rule. Another method often used is Larsen's product operation rule. The resulting fuzzy output MF is a result of superimposing all fired fuzzy output MFs. This process is called composition of output MFs to one output MF.
The rulebase above can also be represented in a Karnaugh-Veitch (KV) type diagram. The following diagram shows all membership levels for the inputs and

(c) 1993-97 FDG-Systems N. Exler

the outputs. Each table entry represents a fuzzy rule.

		T	
	heating	cold	warm
ΔT	negative	high	medium
	zero	high	off
	positive	medium	off

In figure 49 the whole process of fuzzification, inference processing, composition and defuzzification is graphically shown for two different rules (one rule is intentionally different to the rule base above).

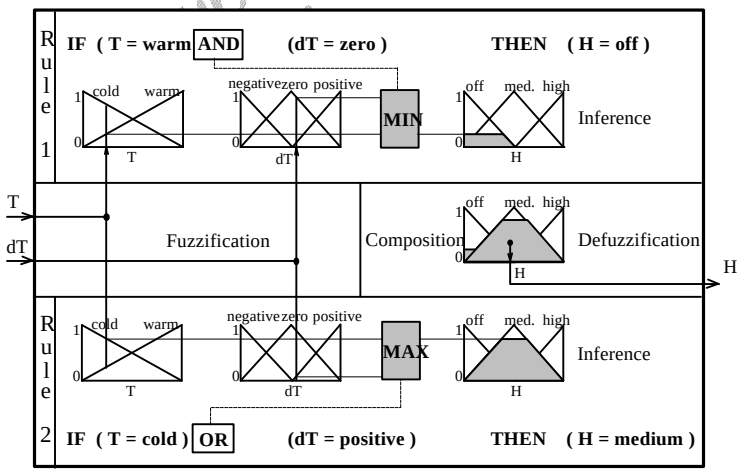


Fig. 49: A fuzzy controller with 2 inputs, 1 output and 2 rules.

7.3.3 Defuzzification

Finally, the defuzzification process converts the fuzzy outputs from the rule evaluation step into crisp system outputs. There are several possible methods of performing the defuzzification. A common defuzzification method, especially for control purposes, is the center of gravity (COG) or centroid defuzzification method.There are also a lot of other popular defuzzification methods like the mean of maximum (MOM), maximum left (ML) or maximum right (MR) defuzzification.

The fuzzy outputs obtained in the inference and composition step are used to truncate the corresponding output membership function by the appropiate truth values as shown in figure 50. Then, the center of gravity of the resulting fuzzy set is found by finding the balance point of the resulting membership function and only the value along the heating axis, which is a result of the projection of the center of gravity to the heating axis, is used as the crisp output (shown in figure 50 with an arrow).

(c) 1993-97 FDG-Systems N. Exler

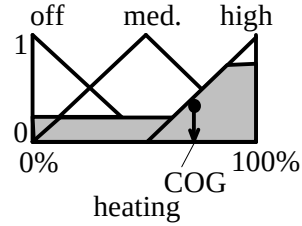


Fig. 50: Center of gravity defuzzification

The result of the defuzzification using other methods than COG is shown in figure 51.

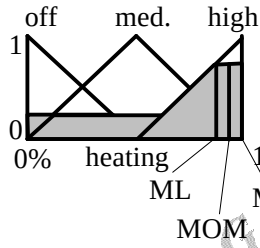


Fig. 51: MOM, ML and MR defuzzification.

(c) 1993-97 FDG-Systems N. Exler

Instead of using discrete membership functions for the output fuzzy sets, singletons are often used for embedded control applications. Then, the following simple computation formula can be used:

$$\text{Centroid} = \sum_{\text{all } x} \mu(x)*x \; / \; \sum_{\text{all } x} \mu(x)$$

where $\mu(x)$ is the fuzzy output value and x are the centroids in the universe of discourse of **X.** For the example mentioned in chapter 2.2, the output becomes:

heating = (0.25 * 0% + 0.25 * 50% + 0.75 * 100%) / (0.25 + 0.25 + 0.75) = 70 %

This crisp output value can be used directly to perform a control action or some other data post processing can be performed.

7.4 Fuzzy Control

7.4.1 Control structure

The standard control loop structure is shown in figur e 52. The ADSP performs all the data pre- and post - processing and the fuzzy c ontrol on a single processor. If it is necessary the fuzzy controller can be combined with classical controllers like PID's to realize mixed contro l solutions with a very high performance at low cost

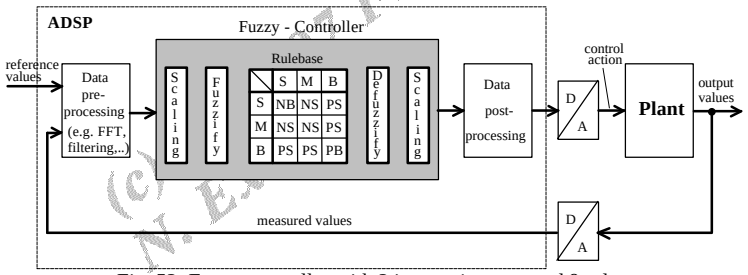


Fig. 52: Fuzzy controller with 2 inputs, 1 output and 9 rules.

7.4.2 Design steps of a fuzzy controller

- Start with a definition of the problem.
  Normally, this is not a mathematical model, just how the system to be controlled should work.
- Definition of all input (which can be measured or calculated) and output variables (linguistic variables) and their input and output ranges.
- Definition of membership functions for each input and output variable.

For the output MFs normally just singletons are used. FID Fuzzy Application Software uses automatically singletons where applicable.

- Establishment of a set of fuzzy rules (rule-base) in the form of IF-THEN rules which describe how the system works.
  Use the process knowledge and the experience of human experts !
- Definition of an inference method
  Normally the MIN-MAX inference is used.
- Definition of a defuzzification method.
  Normally the center of gravity (COG) method.
- Simulate the system and tune the design by modifying the scale values and the membership functions.
- Compile membership functions and rules for target digital signal processor ADSP-21xx.
  This is a task where a development tool with optimized code generation really enhances the performance of the fuzzy solution.
- Add pre-processing and post-processing code to complete the application.
  Conventional and fuzzy approaches can coexist on an ADSP-21xx for very flexible, high dynamics and low cost controls.
- Download and test your code with a hardware development system and your target application. (like EZ-Lab, EZ-Teach, Fuzzy EZ-Teach, ..)

(c) 1993-97 FDG-Systems N. Exler

8. Bibliography

Altrock v. C., Über den Daumen gepeilt, c't, (1991),.Heft 3, S. 188 - 200.

Gupta M.M., and Yamakawa T., Fuzzy Computing Theory, Hardware, and Appplications, 1988, Amsterdam, New York, Oxford, Tokyo.

Gupta M.M. and Qi J., Theory of T-norms and fuzzy inference methods, Fuzzy Sets and Systems 40 (1991) p. 431-450.

Kickert W.J.M., Mamdani E.J., Analysis of a fuzzy logic controller, Fuzzy Set and Systems 1, (1978), p. 29- 44.

Koch M., Kuhn T., Wernstedt J., Ein neues Entwurfskonzept für Fuzzy-Regelungen, (1993), at- Automatisierungstechnik 41 (1993) 5, Oldenburg Verlag.

Kosko B., Neural Networks and Fuzzy Systems, 1992, Prentice-Hall International Inc..

Mamdani E.H. and Assilian S., A case study on the application of fuzzy set theory to automatic control, Proc. IFAC Stochastic Control Symp., Budapest (1974).

McNeill D., Freiberger P.,Fuzzy Logic - The revolutionary computer technology that is changing our world., (1993), A Touchstone book

Mizumoto M., Fukami S. and Tanaka K., Some methods of fuzzy reasoning, in: M.M. Gupta et al., Eds., Advances in Fuzzy Set Theory and Applications, North-Holland, Amsterdam, (1979), p. 117-136.

Pedrycz W., Fuzzy Control and Fuzzy Systems, (1989), John Wiley & Sons Inc..

Raju G.V.S., Zhou J., Kisner R.A., Hierarchical fuzzy control, Int. J. Control, Vol. 54, (1991), 5, p. 1201- 1216.

Sugeno M., Ed., Industrial Applications of Fuzzy Control, North-Holland, Amsterdam, 1985.

Tilli T., Fuzzy Logik, Grundlagen, Anwendungen, Hard- und Software, (1991), München: Franzis Verlag.

(c) (1993-97 FDG-Systems N. Exler

Zadeh L.A., Fuzzy Sets, Information and Control (1965), p. 338
Zadeh L.A., Outline of a new approach to the analysis of complex systems and decision process, IEEE Trans. Systems Man Cybernet. 3 (1973) p. 28-44.
Zimmermann H.-J., Fuzzy Sets Theory - and its Applications, 2nd Ed. 1991, Kluwer Academic Publishers.

(c) 1993-97 FDG-Systems
N. Exler

Index

ADSP source-code
- extension 17
- features 56
- fuzzy 54
- generation 26, 39
- implementation 77
- KV type 54

Controller
- design 132
- examples 85, 94, 100, 108
- fuzzy 39, 94, 132
- Fuzzy PID 65, 100, 108
- PID 39

Defuzzification 58, 130
- memory model 71
- methods 34, 58, 68, 130

File extension 17
Fuzzification 57, 58, 125
- memory model 61

Fuzzy Logic
- composition 127
- crisp set 115
- defuzzification 130
- fuzzification 125
- fuzzy sets 114, 116
- inference 127
- introduction 113
- membership function 114, 115
- operator 121

Fuzzy map 35
Fuzzy resource file
- extension 17
- status 24

Fuzzy sets
- add 28
- definition 25, 59
- introduction 114
- shapes 26, 116, 119

Fuzzy System
- structure 124

Inference 57, 127
- MAX-MIN 30, 128
- memory model 66, 67
- methods 57, 58, 62, 128
- operator 30, 31

Linguistic variable
- add 27
- definition 25

Plant
- add 40
- Disturbance 50
- examples 42, 43, 45
- extension 17
- type 40-Page: 42, 44

Resolution
- input 75
- membership function 76
- output 76
- scaling 75

Rules
- add 31, 33
- antecedents 57
- consequent 57
- delete 33
- edit 31, 32
- expert type 33
- KV type 31, 87, 102, 110, 129
- no rules 32

(c) 1993-97 FDG-Systems N. Exler

status 30
type 31
Simulation
controller 39
definition 46, 47
Disturbance 44, 50
examples 46, 48, 49
fuzzy 48
fuzzy closed-loop 49
Fuzzy PID 105
PID 112
plant 39
save 52
SDF definition 52, 53
view 51

(c) 1993-97 FDG-Systems
N. Exler

FID Fuzzy Registration Form

Register immediately to receive these benefits:

- Information on FID fuzzy application software
- Information on new Hard- and Software for control applications
- Upgrade Information

Return this form to:

FDG-Systems
Dipl.-Ing. Norbert Exler
Leystrasse 48/25
1200 Vienna, Austria - Europe

First Name

Last Name

Company Name (if applicable)

Job Title

Address

City State

Country ZIP Code

Telephone Number, Fax Number (include Area Code!)

eMail Address

FID Fuzzy Application Software Serial Number

If you have any suggestions or comments concerning the manual or software, please write th em on the reverse of this form, or send us an eMail!